<http://www.diva-portal.org>

This is the published version of a paper presented at *2017 International Conference on Machine Learning and Soft Computing, ICMLSC, Ho Chi Minh City*.

Citation for the original published paper:

Pham, P., Erlandsson, F., Wu, F. (2017)

Social Coordinates: A Scalable Embedding Framework for Online Social Networks.

In: *Proceedings of the 2017 International Conference on Machine Learning and Soft Computing* (pp. 191-196). ACM Press

<https://doi.org/10.1145/3036290.3036298>

N.B. When citing this work, cite the original published paper.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:bth-14116>

Social Coordinates: A Scalable Embedding Framework for Online Social Networks

Phuong Pham
UC Davis
pdpham@ucdavis.edu

Fredrik Erlandsson
Blekinge Institute of
Technology
fredrik.erlandsson@bth.se

S.Felix Wu
UC Davis
wu@cs.ucdavis.edu

ABSTRACT

We present a scalable framework to embed nodes of a large social network into an Euclidean space such that the proximity between embedded points reflects the similarity between the corresponding graph nodes. Axes of the embedded space are chosen to maximize data variance so that the dimension of the embedded space is a parameter to regulate noise in data. Using recommender system as a benchmark, empirical results show that similarity derived from the embedded coordinates outperforms similarity obtained from the original graph-based measures.

CCS Concepts

• **Human-centered computing**→Collaborative and social computing→Collaborative and social computing design and evaluation methods

Keywords

Embedding, graph kernels, online social networks.

1. INTRODUCTION

The ubiquity of social media leaves an abundant amount of digital traces about interactions between people and their peripheral world. This source of information is valuable in various disciplines and several applications, for example, targeted marketing, political campaigns, match-making, and so on. However, this rich source of information has several challenges. To several machine learning and data analytic problems, the adjacency matrix of the underlying social graph of social media, where nodes represent people and items (news articles, product information, discussions, etc.) and edges represent the relationships between these subjects, is considered as raw data. Some sorts of transformations are needed to make the raw data more usable. The ubiquity of social media, on one hand, indicates their success, but on the other hand it imposes additional challenges on the tools used to deal social network data. These tools must be scalable to cope with the enormous size of the data. In contrast to the social networks that have been well-studied in network theory literature, e.g. bibliographical network, network induced from product reviews, movie ratings, etc., where it costs a significant amount of "investment" (time, effort, money) for an

edge to be formed, in social media, people interact with their peripheral world in a more ad-hoc manner. As a consequence, the issue of noise in social media data needs to be addressed more carefully compared to the other types of network data.

The purpose of doing transformation on adjacency matrix is to extract high level semantics from the raw input data and represent them in a convenient, easy-to-use format. Among several high level semantics that could be derived from the raw adjacency matrix, similarity between graph nodes is not only a fundamental but also a useful metric to extract and preserve. Specifically, several fundamental pattern analysis problems, e.g. clustering, ranking, classification, etc. can be solved by just using pair-wise similarities between data subjects. Despite their potentials, pair-wise similarity based algorithms, commonly referred as kernel methods, have not widely been widely adopted for graph data due to their incapability to scale as the size of input graph grows.

To address the three issues listed above, namely raw data transformation, scalability and noisy data, we propose a novel scalable graph data transformation framework that embeds nodes of the input graph into an Euclidean space such that the pair-wise inner products between data points in the embedded space reflect the pair-wise similarity between the represented graph nodes.

The axes of the embedded Euclidean space are also the principal axes of the data points. Therefore, dimension of the embedded space is a parameter that can be used to control the noise blended in the transformed data.

The scalability of our framework comes from the adoption of landmark-based strategy. Although the computing similarity for each pair of nodes in a (large) graph is prohibitive, the approximation of similarity from one node to the rest of the graph can be done efficiently. Leveraging this approximation, a set of landmark nodes are chosen, and the similarities between these landmarks to the rest of graph are computed. The computed inter-landmark similarities are used to set up the skeleton of the embedding (basis of embedded space). The coordinates of the other ordinary nodes, are then determined based on their relative positions to the landmarks.

2. PRELIMINARIES

2.1 Graph Kernels: Node Similarity Measures

Since similarity between nodes is a fundamental measure to derive from a graph, multiple methods have been proposed to quantify the graph node similarity. These methods can be categorized into two types [11]:

- Neighborhood-based measures which basically take into account only the node neighbor $\Gamma(u)$, the set of nodes adjacent to u . For examples Jaccard index ($\frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$),

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

ICMLSC '17, January 13-16, 2017, Ho Chi Minh City, Viet Nam
© 2017 ACM. ISBN 978-1-4503-4828-7/17/01...\$15.00

DOI: <http://dx.doi.org/10.1145/3036290.3036298>

Adar/Adamic $\sum_{z \in \Gamma(u) \cap \Gamma(v)} \log^{-1} \Gamma(z)$, preferential attachment $|\Gamma(u)||\Gamma(v)|$ [11].

- The other more global category take into account the ensemble of all paths between the node pair. These methods involve with the exponents of the graph adjacency matrix A or exponents of matrices derived from adjacency matrix A , e.g. graph Laplacian, or transition probability matrix. We refer interested readers to [5] for a comprehensive review of graph kernels.

Pair-wise proximity between nodes form a symmetric matrix. If this matrix is positive semidefinite, the measure is referred to as graph kernel. The advantages of graph kernels are: compared to neighborhood-based, they make use more information from the input graph, and, more importantly, proximity measure can be established for all node pairs. In this paper, we focus on two graph kernels that directly make use of adjacency matrix:

von-Neumann diffusion kernel [16]:

$$K_{VNM} = \sum_{k=0}^{\infty} \alpha^k A^k \quad (1)$$

and exponential diffusion kernel [8]:

$$K_{ED} = \sum_{k=0}^{\infty} \frac{\alpha^k A^k}{k!} \quad (2)$$

Note that element A_{ij}^k , and K_{ij} specify the number of k -length walks between node i and j , and proximity between node i and j , respectively. The parameter $0 < \alpha < 1$ controls the decaying significance of the longer paths contributed to the total proximity between the two nodes. Intuitively, the more shorter paths between two nodes, the more similar those nodes are. As shown in section 3, these kernels can be efficiently approximated via a series of input graph's adjacency matrix multiplication. This characteristic makes these kernels compatible with our scalable framework

2.2 Kernel-based Euclidean Embedding

The kernel matrices (K_{NMN}, K_{ED}) are all symmetric positive semi-definite [5]. Let V be a matrix whose columns are K 's orthonormal eigenvectors; $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots)$ where λ_i are K 's eigenvalues, kernel matrix can be factorized:

$$K = V^T \Lambda V = X^T X \text{ (where } X = \Lambda^{\frac{1}{2}} V \text{)}$$

The fact that K can be factorized $K = X^T X$ implies graph nodes can be embedded into an Euclidean space such that the inner product of a pair of vectors is equivalent to the proximity between two corresponding nodes. The factorization of K is, however, not unique. For example, because K is symmetric positive semidefinite, it can also be factorized using Cholesky decomposition method $K = X^T X$ where X is an upper triangular matrix. In the next section, we examine the criteria of choosing an optimal embedding for graph nodes.

2.3 Kernel PCA

Let $\Phi \in R^{d \times n}$ be a matrix whose column Φ_u is the coordinates of node u in some basis (this implies $K = \Phi^T \Phi$). Let $C = \Phi \Phi^T$, projecting columns of Φ onto a small set of orthonormal eigenvectors associated with C 's largest eigenvalues of matrix maximizes data variances, reduces the dimension of embedded space, and provides a means handle noise. Eigenvalues and eigenvectors of C can be computed via kernel matrix K . The

technique is known as Kernel Principal Component Analysis (Kernel PCA) [14], [16].

$$K = \Phi^T \Phi = V^T \Lambda V$$

Now, consider an eigenvector v of matrix K and the multiplication of vector Φv with matrix C :

$$C \Phi v = (\Phi \Phi^T) \Phi v = \Phi (\Phi^T \Phi) v = \Phi K v = \lambda (\Phi v)$$

This means: Φv are eigenvectors of C ; K and C have the same set of eigenvalues. Note that v 's are orthonormal vectors ($v^T v = 1$, normalization of Φv is:

$$\frac{\Phi v}{\|\Phi v\|} = \frac{\Phi v}{\sqrt{v^T \Phi^T \Phi v}} = \frac{\Phi v}{\sqrt{v^T K v}} = \frac{\Phi v}{\sqrt{\lambda v^T v}} = \frac{\Phi v}{\sqrt{\lambda}}$$

Therefore, the coordinate of a node u :

$$P_{V_k}(\Phi_u) = (P_{v_1}(\Phi_u), \dots, P_{v_l}(\Phi_u), \dots, P_{v_k}(\Phi_u)) \quad (3)$$

where:
$$P_{v_i}(\Phi_u) = \frac{\Phi_u^T \Phi v_i}{\sqrt{\lambda_i}} \quad (4)$$

The notations $P_{V_k}(\Phi_u)$ and $P_v(\Phi_u)$ emphasize that the resulting coordinate of a node u is related to the "projection" of Φ_u onto the eigenvectors V_k corresponding to the largest eigenvalues of K . The dimension of result embedded space is determined by the size of V_k . Hereby, we drop the subscript k so that the resulting coordinate of a node u is $P_V(\Phi_u)$.

3. SCALABLE EMBEDDING

It is obvious that even computing proximity between node u and v , i.e. element K_{uv} according to its closed forms (1), (2), is prohibitive, leave alone computing and storing the whole kernel matrix K . To work around these computation and storage costs, we approximate both the element K_{uv} and K .

3.1 Landmark-based Embedding

Note that in (3) and (4), $\Phi_u^T \Phi$ is the proximity from node u to the nodes from which the eigenvalues and eigenvectors are computed. We can approximate coordinate of a node u using a landmark-based approach (the hat notation ($\hat{x}$) means x is approximated):

- First, choose a set of landmarks L . L must be sampled so that they reflects the structure of proximity distribution between nodes of input graph, i.e. the whole kernel matrix K .
- Compute the kernel matrix between these landmarks only, $\hat{K}_L \in R^{|L| \times |L|}$ and determine $\hat{K}_L$'s eigenvectors and eigenvalues.
- The coordinate of a node u is obtained from $\hat{K}_L$'s eigenvectors and eigenvalues ($\hat{V} = \{\hat{v}_i\}, \{\hat{\lambda}_i\}$) and proximity from u to the landmark nodes L ($\hat{R}_u = \hat{\Phi}_u^T \hat{\Phi}$). (Note that $\hat{R}_u$ is a row of $\hat{R} \in R^{n \times |L|}$, the proximity from all nodes to the landmarks, computation of $\hat{R}$ is presented in section (3.3)):

$$\begin{aligned} \hat{P}_V(\hat{\Phi}_u) &= \left(\dots, \frac{\hat{\Phi}_u^T \hat{\Phi} v}{\sqrt{\hat{\lambda}}}, \dots \right) = \left(\dots, \frac{\hat{R}_u v}{\sqrt{\hat{\lambda}}}, \dots \right) \\ &= (\dots, P_v(\hat{\Phi}_u), \dots) \end{aligned} \quad (5)$$

3.2 Relationship with Nystrom method:

Equation (5) can be expressed in closed form:

$$P_V(\hat{\Phi}_u) = \hat{\Phi}_u^T \hat{\Phi} V \Lambda^{-1/2} = \hat{R}_u V \Lambda^{-1/2}$$

Coordinates of all nodes in the input graph:

$$P_{\tilde{v}}(\tilde{\Phi}) = \tilde{R}V\Lambda^{-1/2}$$

The pair-wise proximity derived from embedded coordinates:

$$\begin{aligned}\mathcal{K} &= P_{\tilde{v}}(\tilde{\Phi})P_{\tilde{v}}(\tilde{\Phi})^T \\ &= \tilde{R}V\Lambda^{-1}V\tilde{R}^T \approx K\end{aligned}\quad (6)$$

Equation (6) is known as Nystrom method to approximate a kernel matrix. The development presented in this paper provides another interpretation for Nystrom method, which was motivated by a numerical solution for integral equation [19].

3.3 Proximity Approximation

Embedding a node u of a graph is now reduced to computing the pair-wise proximities between landmark nodes (i.e. calculating $\tilde{K}_L$) and proximities between u and landmark nodes L , $\tilde{K}_u$. To embed all nodes, the only obstacle is to compute the proximity from each landmark to the rest of the graph.

Since the contribution of longer paths toward the proximity between a pair of nodes are negligible, we can efficiently approximate the proximity by just discarding the paths whose lengths exceed a certain threshold ℓ_{max} . Let $A_L \in R^{n \times |L|}$ be the columns of the adjacency matrix A corresponding to the landmark nodes, the number of paths from the landmarks to the rest of the graph of length m is $A^{m-1}A_L$. The approximated proximity from the landmarks to the rest of nodes, $\tilde{K} \in R^{n \times |L|}$, are given by:

$$\tilde{K} = \sum_{i=0}^{\ell_{max}-1} \alpha^i A^i A_L \quad (7)$$

$\tilde{K}_L$ is obtained by extracting from $\tilde{K}$ the rows corresponding to the landmarks L . Due to the sparsity of the adjacency matrix A , the complexity of this computation is $O(|L||E|\ell_{max})$ where E is the set of edges of the input graph.

Approximation for exponential diffusion kernel can be done similarly:

$$\tilde{K}_{ED} = \sum_{i=0}^{\ell_{max}-1} \frac{\alpha^i A^i A_L}{i!} \quad (8)$$

3.4 Normalizing Embedded Vectors

The proximity between two nodes u, w is approximated by the inner product of the two vectors $u = P_{\tilde{v}}(\tilde{\Phi}_u)$ and $w = P_{\tilde{v}}(\tilde{\Phi}_w)$, i.e. $\langle u, w \rangle \geq |u||w|\cos(u, w)$. To remove the dependency on vector norms, nodes are embedded on a hypersphere of unit radius. This constraint can be enforced by normalizing the proximity matrix K :

$$\bar{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii}}\sqrt{K_{jj}}} \quad (9)$$

This ensures $\forall i, j: \bar{K}_{ij} \leq \bar{K}_{ii} = 1$ and $\bar{K}_{ij} \leq \bar{K}_{jj} = 1$.

However, for $u \notin L$, $\tilde{K}_{uu}$ is not available in $\tilde{K}$. Therefore, we cannot normalized $\tilde{K}_{tu}$ as:

$$\tilde{K}_{tu} = \frac{\tilde{K}_{tu}}{\sqrt{\tilde{K}_{tu}}\sqrt{\tilde{K}_{uu}}} \quad (10)$$

(Here, the notation $\sim$ denotes the normalization of an approximated amount)

Fortunately, recall that according to (5), the normalized coordinate of an ordinary node u is the vector:

$$\begin{aligned}P_{\tilde{v}}(\tilde{\Phi}_u) &= (\dots, P_{\tilde{v}}(\tilde{\Phi}_u), \dots) \\ \text{where: } P_{\tilde{v}}(\tilde{\Phi}_u) &= \frac{\tilde{K}_{uL}\tilde{v}}{\sqrt{\tilde{\lambda}}} = \frac{\tilde{K}_u\tilde{D}_L\tilde{v}}{\sqrt{\tilde{K}_{uu}\tilde{\lambda}}} \\ \text{and: } \tilde{D}_L &= \text{diag}(\tilde{K}_L)^{-1/2} \\ \text{then: } P_{\tilde{v}}(\tilde{\Phi}_u) &= (\dots, \frac{\tilde{K}_u\tilde{D}_L\tilde{v}}{\sqrt{\tilde{K}_{uu}\tilde{\lambda}}}, \dots) \\ &= \frac{1}{\sqrt{\tilde{K}_{uu}}} (\dots, \frac{\tilde{K}_u\tilde{D}_L\tilde{v}}{\sqrt{\tilde{\lambda}}}, \dots)\end{aligned}$$

On the other hand, since $P_{\tilde{v}}(\tilde{\Phi}_u)$ has unit norm:

$$P_{\tilde{v}}(\tilde{\Phi}_u) = \frac{P_{\tilde{v}}(\tilde{\Phi}_u)}{|P_{\tilde{v}}(\tilde{\Phi}_u)|} = \frac{(\dots, \frac{\tilde{K}_u\tilde{D}_L\tilde{v}}{\sqrt{\tilde{\lambda}}}, \dots)}{|\dots, \frac{\tilde{K}_u\tilde{D}_L\tilde{v}}{\sqrt{\tilde{\lambda}}}, \dots|} \quad (11)$$

Equation (11) means normalization of an embedded vector for ordinary node u does not require the proximity from u to itself.

3.5 Embeddability of Graph Kernels

The outcome of our embedding framework is a low rank approximation of the kernel matrix, i.e. $P_{\tilde{v}}(\tilde{\Phi})^T P_{\tilde{v}}(\tilde{\Phi}) \approx K$. However, small values of decay parameter α in (1),(2),(7) and (8) makes the kernel matrix K resemble an identity matrix, i.e. $K_{ii} \gg K_{ij} \forall j \neq i$, and prevents K from having a useful low rank approximation.

On the other hand, K can be considered as the sum of rank-one matrices (outer products of eigenvectors) weighted by the eigenvalues.

$$K = \sum_i \lambda_i v_i v_i^T$$

The best rank- k approximation of K is obtained by keeping only the k largest terms. This means K is easily approximated by a low rank matrix if its eigenvalues are skewed, i.e. some significantly larger than the others.

For von-Neumann and exponential diffusion kernels, a simple yet efficient way to improve their embeddability while maintaining its symmetric positive semi-definiteness is uniformly offsetting a constant equal to the minimum eigenvalue of the kernel matrix from the on-diagonal elements.

$$K^* = K - \lambda_{min}I \quad (12)$$

where λ_{min} is the smallest eigenvalue of K .

This adjustment turns out to be a general form of Katz measure [7]:

$$Katz = \sum_{k=1}^{\infty} \alpha A = K_{NMN} - I$$

Our scalable embedding nodes of a graph into a hypersphere of unit radius can be summarized as follows:

Input:

- A : adjacency matrix of graph $G(\mathcal{U} \cup \mathcal{V}, E)$
- k : dimension of embedded space

Output:

Matrix $P_{\tilde{V}}(\Phi) \in R^{k \times n}$, $n = |\mathcal{U} \cup \mathcal{V}|$, column $P_{\tilde{V}}(\Phi_u)$ is the coordinate of node u

Select a set of landmark L from $\mathcal{U} \cup \mathcal{V}$

Compute $\tilde{K} \in R^{n \times |L|}$, approximated proximity from nodes in G to L

Extract $\hat{K}_L \in R^{|L| \times |L|}$, approximated proximity between nodes in L from $\tilde{K}$

Adjust $\hat{K}_L$'s diagonal: $\tilde{K}_L = \hat{K}_L - \lambda_{\min} I$

Normalize $\tilde{K}_L$: $\tilde{K}_L = \text{diag}(\tilde{K}_L)^{-1/2} \tilde{K}_L \text{diag}(\tilde{K}_L)^{-1/2}$

Find eigenvectors $\tilde{V} = \{\tilde{v}_i\}$, eigenvalues $\{\tilde{\lambda}_i\}$ of $\tilde{K}_L$

Pick k largest $\tilde{\lambda}$'s and the corresponding $\tilde{v}$. Each column $P_{\tilde{V}}(\Phi_u)$ is:

$$P_{\tilde{V}}(\Phi_u) = \frac{(\dots, \frac{\tilde{K}_u \tilde{D}_L \tilde{v}}{\sqrt{\tilde{\lambda}}}, \dots)}{|\dots, \frac{\tilde{K}_u \tilde{D}_L \tilde{v}}{\sqrt{\tilde{\lambda}}}, \dots|} \text{ where } \tilde{D}_L = \text{diag}(\tilde{K}_L)^{-1/2}$$

Algorithm 1 Scalable embedding algorithm

4. EXPERIMENTS

4.1 Dataset

To evaluate the embedding framework, we use data about discussions in public Facebook pages [4]. Each Facebook page has a set of posts, under each post, people show their opinions by either liking the post, making comments, or liking comments of others. We use data from three popular news media pages, i.e. The New York Times, CNN and Fox News with the posts in year 2012. For each page, we extract an undirected, unweighted, bipartite graph $G = (U \cup V; E)$ where U and V are sets of users and posts, an edge $e = (u; w)$ is used to indicate that the user u interacted with the post w (by either making a comment, liking the post, or liking a comment under the post).

Below are the details about our data set:

	Users	Posts	Edges
NYTimes	1,354,780	3,134	3,561,465
CNN	1,882,876	100,773	5,628,966
Fox News	1,045,283	1,559	6,117,852

We then filter out the users who have low activity or extremely active (those has number of activity less than 20 or more than 200) and only keep the largest connected component

	Users	Posts	Edges
NYTimes	19,010	3,134	793,410
CNN	36,453	60,749	1,560,665
Fox News	57,559	1,557	2,588,088

4.2 Similarity Measure

Examining similarity quantification performance is central to our experiments and due to the unavailability of groundtruth data about similarity between users, and similarity between posts, we

evaluate similarity quantification of our embedding framework in one of its inherent application, neighborhood-based recommender system.

A neighborhood-based recommender system determines how well an item i matches the interests of a customer u by considering either: *i*) how users u' that are similar to u response to product i *ii*) how user u responses to the products i' that are similar to i . Given a similarity measure between user u and other users u' $Sim(u; u')$ or between item i and other items i' $Sim(i; i')$, predicted response of user u for item i is:

$$\hat{R}(u, i) = \frac{\sum_{u'} Sim(u, u') R(u', i)}{\sum_{u'} Sim(u, u')}$$

$$\text{or: } \hat{R}(u, i) = \frac{\sum_{i'} Sim(i, i') R(u, i')}{\sum_{i'} Sim(i, i')}$$

We compare similarity measures by a kernel $Sim(u, u') = K_{uu'}$, or by the corresponding embedded space: $Sim(u, u') = P_{\tilde{V}}(\Phi_u)^T P_{\tilde{V}}(\Phi_{u'})$.

To evaluate similarity quantification of a measure or a parameter setting of a measure: a set of user nodes are randomly selected, for each selected user node, we put aside a set of edges related to that user node to form the Test data. The remaining graph is used as Training data. Then, the similarity measure obtained from using Training data is plugged into the recommender system to make a recommendation of the most relevant items. The accuracy of this recommendation is measured by the recall score.

$$recall = \frac{T_p}{T_p + F_n}$$

where T_p is the number recommended items actually in the Test set; F_n is the number of recommended items but do not belong to the Test set; none of the *(user, recommended item)* is in the Training set.

Unless explicitly stated otherwise, we use the following parameters: maximum path length ℓ_{max} parameter in (7), (8) is 6; number of test users (randomly selected) is 100; for each test user, number of edges reserved for Test data is 10%; number of nearest neighbors is 10; number of recommended items is 10. Each experiment is repeated 30 times.

We first investigate the role of decay parameter α by measuring the performance of the original graph kernels while varying the value of α . Figure 1 shows that smaller values of α tend to yield better performance; for $\alpha < .001$, the performances become stable.

Fixing $\alpha = .001$ and varying numbers of landmarks and dimensions, we compare the similarity quantification performance between the original graph kernels and its corresponding embedded versions. Figure 2 and Figure 3 show the normalized difference between two methods which is measured as:

$$\frac{recall(\text{coordinate}) - recall(\text{kernel})}{recall(\text{kernel})} \times 100\%$$

We can see that: with sufficient landmarks and dimension, the proximity derived from embedded coordinates outperforms the original kernels. The improvment could be attributed to the denoise ability of embedded coordinates.

Figure 1 Effect of decay parameter α on original graph kernels

Figure 2 Coordinate vs. Original graph kernels while varying number of landmarks(#dimensions=80%landmarks)

Figure 3 Coordinate vs. Original graph kernels while varying dimension of embedded space (#landmarks: 800)

4.3 Embeddability

As mentioned in Section 3.5, the skewness of K 's eigenvalues is an indicator of the embeddability of a kernel. To compare the embeddability of an original kernel $\tilde{K}$ with its adjusted version $\tilde{K}^*$, we obtain their eigenvalues, sort the eigenvalues descendingly, and calculate the accumulated sum.

Figure 4 shows the accumulated sum of the adjusted and original kernels, we can see that the original kernels yield almost identical

Figure 4 Skewness of eigenvalues of original and adjusted kernels eigenvectors while the eigenvectors resulted from the adjusted kernels are more skewed.

5. RELATED WORK

5.1 Kernel methods

Kernel methods bridges the gap between the traditional tools dealing with linear input and the nonlinear relationship of features in reality. Compared to other machine learning techniques (e.g.

neural networks), kernel methods have solid mathematical foundation and can be used to address several problems in machine learning, e.g. regression, classification, ranking, clustering, canonical correlation analysis. Comprehensive treatments of kernel methods can be found in [15], [16].

Multidimensional scaling [2], [1] (MDS) is a family of methods to embed data entities into an Euclidean space. MDS take pairwise dissimilarity (δ_{ij}) as input and find embedding whose interpoint distance reflect the dissimilarity. The embedded inter-point distances are made to approximate the input dissimilarity explicitly, as in classical MDS, or approximate the transformed dissimilarity, as in metric MDS.

[14] introduced kernel PCA method to identify directions where variances of [latent] feature space are maximized given the kernel matrix of pairwise similarity between data entities. [20] pointed out the connection between MDS and kernel PCA.

5.2 Nystrom method

As mentioned in Section 3.2, our embedding is similar to and provides an intuitive interpretation for Nystrom method, a low-rank approximation of kernel matrix. Nystrom method was first proposed in [19] in which the landmarks were uniformly sampled. [3] studied theoretical bounds for the approximation when landmarks were sampled uniformly and sampled according to non-uniform probability distribution

5.3 Graph kernels

von-Neumann and Exponential Diffusion Kernel are among several node similarity measures one can derive from a graph. Other graph kernels were constructed based derivatives of graph adjacency matrix, e.g. graph Laplacian, transitionprobability matrix [5], [8], [17], [6]. A throughout summary of graph kernels can be found in [5].

To approximate proximity from a node (landmark node) to the others, we do a series of adjacency matrix multiplication. This method was suggested in [12]. [18], [11] applied this technique to compute Katz, Rooted PageRank and Escape probability proximity for nodes in large social networks.

5.4 Other Euclidean embedding approaches

Ng et al. [13] proposed Global Network Positioning mechanism to embed computer hosts on Internet to a low-dimensional Euclidean space. They first measured (ground-truth) "ping delay" from a set of landmark nodes to each other and to other ordinary targets. Coordinates of landmarks, and then ordinary nodes were determined by minimizing cost functions of the differences between inter-landmark distances (coordinate-based and ground-truth) and differences between landmark-ordinary distances.

In collaborative recommender system community, latent factor models [9] [10] also profile users and items with vectors. The bipartite matrix B (size: users-by-items) of the users' rating was factorized to the latent feature vectors of users P (size: users-by-features) and items Q (size: items-by-features).

6. CONCLUDING REMARKS

We have provided details about a scalable framework to embed nodes of a large social network into Euclidean space. The inner products between embedded data points reflect the inter-node proximity derived from the input graph using the compatible kernels. We first approximately identify the skeleton of the

embeddings, i.e. the basis of embedded space, via a set of landmark nodes. The basis is the eigenvectors corresponding to the largest eigenvalues of the landmarks-only kernel matrix. Then, coordinate of an ordinary node is derived from its proximity to the landmarks.

The problem is reduced to computing the the proximity from a landmark to the rest of the graph (to other landmarks and ordinary nodes). This computation can be efficiently approximated by taking advantage of the sparse adjacency matrix.

Dimension of the embedded space, or number of principal axes, is a parameter of the framework that can be used to manipulate the effect of noisy data. Empirical results show that our framework improve the performance of similarity quantification in neighborhood-based recommender system application.

7. REFERENCES

- [1] I. Borg and P. J. Groenen. Modern multidimensional scaling: Theory and applications. Springer Science & Business Media, 2005.
- [2] M. A. Cox and T. F. Cox. Multidimensional scaling. In *Handbook of data visualization*, pages 315-347. Springer, 2008.
- [3] P. Drineas and M. W. Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(Dec):2153-2175, 2005.
- [4] F. Erlandsson, R. Nia, M. Boldt, H. Johnson, and S. F. Wu. Crawling online social networks. In *Network Intelligence Conference (ENIC), 2015 Second European*, pages 9-16. IEEE, 2015.
- [5] F. Fouss, K. Francoise, L. Yen, A. Pirotte, and M. Saelens. An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification. *Neural networks*, 31:53-72, 2012.
- [6] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saelens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering*, 19(3):355-369, 2007.
- [7] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39-43, 1953.
- [8] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *ICML*, volume 2, pages 315-322, 2002.
- [9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426-434. ACM, 2008.
- [10] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 145186. Springer, 2011.
- [11] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019-1031, 2007.
- [12] M. Newman. *Networks: an introduction*. Oxford university press, 2010.
- [13] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *INFOCOM*, 2002.
- [14] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583-588. Springer, 1997.
- [15] B. Schölkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.
- [16] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge university press, 2004.
- [17] A. J. Smola and R. Kondor. Kernels and regularization on graphs. In *Learning theory and kernel machines*, pages 144-158. Springer, 2003.
- [18] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in online social networks. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 322-335. ACM, 2009.
- [19] C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In *Proceedings of the 14th annual conference on neural information processing systems*, number EPFL-CONF-161322, pages 682-688, 2001.
- [20] C. K. Williams. On a connection between kernel pca and metric multidimensional scaling. *Machine Learning*, 46(1-3):11-19, 2002.