Geometric Matrix Completion via Sylvester Multi-Graph Neural Network

Boxin Du
boxin@amazon.com
Amazon
New York, USA

Changhe Yuan
ychanghe@amazon.com
Amazon
New York, USA

Fei Wang
feiww@amazon.com
Amazon
San Francisco, USA

Hanghang Tong
htong@illinois.edu
University of Illinois at Urbana Champaign
Urbana, Illinois, USA

ABSTRACT

Despite the success of the Sylvester equation empowered methods on various graph mining applications, such as semi-supervised label learning and network alignment, there also exists several limitations. The Sylvester equation's inability of modeling non-linear relations and the inflexibility of tuning towards different tasks restrict its performance. In this paper, we propose an end-to-end neural framework, SyMGNN which consists of a multi-network neural aggregation module and a prior multi-network association incorporation learning module. The proposed framework inherits the key ideas of the Sylvester equation, and meanwhile generalizes it to overcome aforementioned limitations. Empirical evaluations on real-world datasets show that the instantiations of SyMGNN overall outperform the baselines in geometric matrix completion task, and its low-rank instantiation could further reduce the memory consumption by 16.98% on average.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Information systems** → **Social recommendation**; *Recommender systems.*

KEYWORDS

matrix completion; Sylvester equation; Graph Neural Networks

ACM Reference Format:
Boxin Du, Changhe Yuan, Fei Wang, and Hanghang Tong. 2023. Geometric Matrix Completion via Sylvester Multi-Graph Neural Network. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3615170

1 INTRODUCTION

The Sylvester equation plays a central role for various applications in applied mathematics [10] [18], systems and control theory [2], machine learning [1] and graph mining [12]. Particularly in graph mining, the Sylvester equation has shown its applicability in numerous multi-network mining tasks, such as network alignment [23], social recommendation [6], and semi-supervised learning [4].

Despite its succinct mathematical formulation, elegant theoretical properties, and numerous efficient solvers, there are several limitations when Sylvester equation is applied on multi-network mining. First, the real-world network data contains various heterogeneous features. However, it is non-trivial to directly incorporate these features of the networks into the classic Sylvester equation formulation. Second, in the task of multi-network association, the classic Sylvester equation essentially calculates a linear transformation from the observed prior multi-network association matrix. However, the non-linear relation between the prior knowledge and the final solution can not be captured by the classic Sylvester equation. Third, the Sylvester equation solver is often independent from the downstream task learning in many graph mining problems, and thus the solution of the Sylvester equation has to be further adapted towards different multi-network mining tasks. For example, in network alignment, the solution of multi-network node association is first calculated. Then, either soft or hard alignment method is conducted as an extra post-processing step, such as greedy match [23]. The equation can not be trained or tuned end-to-end as modern neural networks, and consequently the performance of the downstream tasks might be suboptimal. A natural question is: *How can we get the best of both the traditional Sylvester equation formulation and the neural network models?*

In this paper, we propose a Sylvester Multi-Graph Neural Network framework (SyMGNN) in order to generalize the traditional linear Sylvester equation towards an end-to-end neural network model. Specifically, we focus on geometric matrix completion task, and elucidate two instantiations for the SyMGNN framework. Our proposed approach bears three distinctive advantages compared with both the Sylvester equation and the existing neural models targeted on geometric matrix completion. First, the proposed framework is a general form. It is able to incorporate network features, and flexible to be instantiated towards different downstream tasks. Second, the instantiations of the proposed framework could be trained end-to-end, which directly adapt the solution generation module to the downstream prediction module. Third, for geometric matrix completion, two instantiations are provided based on explicitly learning multi-network association, and learning low-rank representations

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615170

for different input networks, respectively. The low-rank instantiation approach further reduces the model's space complexity.

Related Works. Generally, multi-network mining techniques can be categorized into traditional numerical approaches and recent neural techniques. For numerical methods, *GT-COPR* by Li et al. [13] aims at inferring multi-relations among the entities across multiple networks by a tensor-based optimization method. After that, Li et al. [12] propose an optimization method and a low-rank tensor-based label propagation algorithm for multi-relation inference. Later, a cross-network multi-relation association learning method is proposed in *CGRL* [14]. The traditional Sylvester equation is widely adopted in solving network alignment [5, 24, 27], cross-network similarity learning [6, 9, 11], subgraph matching [8, 15], social recommendation [17], and adversarial attack [25, 26].

2 PROBLEM DEFINITION

Before giving the definition of GNN-based neural Sylvester equation in Section 3, we first provide some preliminaries on the traditional Sylvester equation and the Graph Neural Networks, followed by a formal definition of the geometric matrix completion.

A - Sylvester Equation for Multi-network Mining. Given two networks represented as $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{F}_1\}$, $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{F}_2\}$, and an anchor multi-network association matrix $\mathbf{H}$, which denotes the prior knowledge of the multi-network node associations. The Sylvester equation for multi-network mining is defined as follows [7]:

$$\mathbf{X} = \alpha \tilde{\mathbf{A}}_2 \mathbf{X} \tilde{\mathbf{A}}_1^\top + (1-\alpha)\mathbf{H} \tag{1}$$

where $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{A}}_2$ are the symmetrically normalized adjacency matrices of the input networks. The $\mathbf{X}$ represents the cross-network node association scores which the equation aims to calculate. The scalar $\alpha \in (0, 1)$ is aimed at weighting the multi-network association aggregation term (i.e. $\tilde{\mathbf{A}}_2 \mathbf{X} \tilde{\mathbf{A}}_1^\top$), and the prior knowledge term ($\mathbf{H}$). Due to the normalization of $\mathbf{A}_1$ and $\mathbf{A}_2$, the corresponding linear system of Eq. (1) contains a positive semi-definite coefficient matrix, which guarantees the existence of unique solution for Eq. (1). Solving Eq. (1) is often time-consuming. A straightforward iterative method to solve Eq. (1) is the fixed point iteration. More efficient method is proposed in [7] with linear time and space complexity.

The formulation of this equation for multi-network mining enjoys several distinctive advantages. Firstly, theoretically the existence and uniqueness of the solution $\mathbf{X}$ can be guaranteed. Furthermore, there exists various efficient solvers for the solution. Secondly, the solution $\mathbf{X}$ can be seen as a fixed point of the equation, and can be obtained by iteratively evaluating Eq. (1). Compared to existing neural models, which might contain a number of hidden layers, there is no need to save the hidden states/representations.

However, despite the advantages and effectiveness in various tasks, generally there are also several limitations. Firstly, the numerical features of the nodes can not be effectively utilized for calculating $\mathbf{X}$. Secondly, the $\mathbf{X}$ can be seen as a linear transformation from the prior knowledge matrix $\mathbf{H}$. However, the potential non-linear relationship between them can not be captured by this formulation. Thirdly, the equation is not learnable and not tunable towards a given supervised downstream task.

B - Graph Neural Networks. The Graph Neural Networks (GNN) are powerful deep learning models for network data. The basic idea of GNN model is to learn node representations via learnable aggregation, in which the node features are accumulated and transformed from the neighborhood features. Given a network $\mathcal{G} = (\mathbf{A}, \mathbf{F})$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of $\mathcal{G}$, and $\mathbf{F} \in \mathbb{R}^{n \times d}$ is the feature matrix with d being the dimension of features, representative GNN aggregation at the t-th layer can be written as follows.

$$\mathbf{X}^{(t+1)} = \phi(\tilde{\mathbf{A}}\mathbf{X}^{(t)}\mathbf{W} + \Omega^{(t)}\mathbf{F}) \tag{2}$$

where $\tilde{\mathbf{A}}$ is the normalized adjacency matrix with added self-loops. $\mathbf{W}$ is a learnable parameter matrix for the aggregated features.

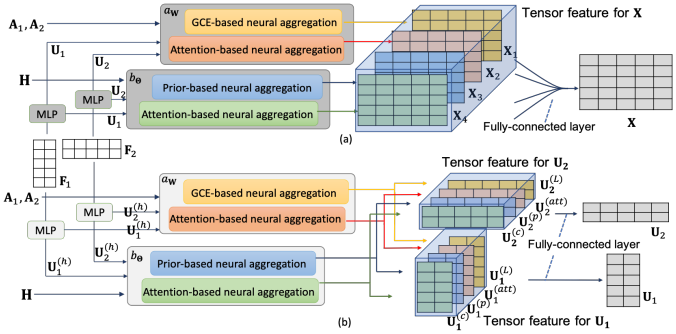

Figure 1: The overall illustration of two instantiations for SyMGNN. (a): the base model, and (b): the low-rank model.

C - Problem Definition. The geometric matrix completion needs to handle two networks which reflect the topological relations between the nodes of two entity sets.

Problem 1. *Geometric Matrix Completion*

Given: *Two networks with node features* $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{F}_1\}$, $\mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{F}_2\}$, *and the partially observed multi-network association* $\mathbf{H}$ *of the nodes in* $\mathcal{G}_1$ *and* $\mathcal{G}_2$*;*

Output: *The unobserved entries in* $\mathbf{H}$.

3 PROPOSED MODEL

3.1 SyMGNN Framework

The goal of the proposed SyMGNN framework is to leverage the advantages of the traditional Sylvester equation, and in the meanwhile overcoming its limitations. First, if we observe the Sylvester equation in Eq. (1) from an iterative perspective, we can see that the first term on the right side aggregate the multi-network node association $\mathbf{X}$ linearly for the updated $\mathbf{X}$. The second term incorporates the prior multi-network association message $\mathbf{H}$ into the updated $\mathbf{X}$. Second, similar to the ideas of the traditional Sylvester equation, we identify the two key modules of the SyMGNN: (1) the multi-network aggregation learning module; and (2) the prior multi-network association incorporation learning module. The general framework can be represented in Eq. (3).

$$\mathbf{X} = \phi(\alpha \cdot a_{\mathbf{W}}(\mathbf{F}_1, \mathbf{F}_2, \tilde{\mathbf{A}}_1, \tilde{\mathbf{A}}_2) + (1-\alpha) \cdot b_{\mathbf{\Theta}}(\mathbf{F}_1, \mathbf{F}_2, \mathbf{H})) \tag{3}$$

where $a_{\mathbf{W}}()$ and $b_{\mathbf{\Theta}}()$ are two neural modules with parameters $\mathbf{W}$ and $\mathbf{\Theta}$, and weighting scalar $\alpha \in [0, 1]$. $\phi()$ is a non-linear activation function. $\mathbf{X}$ is the multi-network association output of the SyMGNN framework, and it can be further fed into a neural network for adapting towards a downstream task in an end-to-end fashion. As we can see, this framework is a neural generalization originated from the Sylvester equation in Eq. (1). When the neural modules $a_{\mathbf{W}}$ and $b_{\mathbf{\Theta}}$ are linear aggregation functions, the Eq. (3) degenerates to the classic Sylvester equation Eq. (1). Numerous instantiations exist for different downstream tasks. Next, we will discuss how to instantiate Eq. (3) for geometric matrix completion.

3.2 Base Model

In order to instantiate $a_{\mathbf{W}}(\mathbf{F}_1, \mathbf{F}_2, \mathbf{A}_1, \mathbf{A}_2)$, we design two parallel layers which adopt the Convolutional Graph Embedding (CGE) aggregation layer and the attention-based aggregation layer respectively [19]. The motivation here is to learn the 2-d hidden representations for the multi-network association solution. In order to achieve this, we design two types of 2-d convolutional non-linear aggregation module, namely the adjacency matrix-based GCE neural aggregation, and the attention-based neural aggregation. To be specific, given $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{F}_1\}, \mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{F}_2\}$ with n_1, n_2 nodes respectively, we first apply the learnable parameters of self-connections on $\mathbf{A}_1$ and $\mathbf{A}_2$ to obtain the updated adjacency matrices. The goal is to adopt the learnable weight of the self-connections from CGE for improving the expressiveness of the model:

$$\hat{\mathbf{A}}_1 = \text{diag}(\boldsymbol{\sigma_1}) + (\mathbf{I} - \text{diag}(\boldsymbol{\sigma_1}))\tilde{\mathbf{A}}_1 \tag{4a}$$

$$\hat{\mathbf{A}}_2 = \text{diag}(\boldsymbol{\sigma_2}) + (\mathbf{I} - \text{diag}(\boldsymbol{\sigma_2}))\tilde{\mathbf{A}}_2 \tag{4b}$$

where $\boldsymbol{\sigma_1}$ and $\boldsymbol{\sigma_2}$ are learnable weights for self-connections of the first network and the second network respectively. Before applying the multi-network aggregation layers, the node features of $\mathcal{G}_1$ and $\mathcal{G}_2$ are fed into an MLP layer for obtaining the hidden features $\mathbf{U}_1 = \text{MLP}_1(\mathbf{F}_1)$ and $\mathbf{U}_2 = \text{MLP}_2(\mathbf{F}_2)$. In the CGE-based multi-network aggregation, the output of the l-th level aggregation is:

$$\mathbf{X}_1^{(l)} = \sum_{i=1}^{l} \phi(\hat{\mathbf{A}}_1^i \mathbf{U}_1 \mathbf{W}_i^{(c)} \mathbf{U}_2^\top (\hat{\mathbf{A}}_2^i)^\top) \tag{5}$$

$\mathbf{W}_1^{(c)}, \cdots, \mathbf{W}_l^{(c)}$ are parameter matrices and $\phi()$ is an activation function. In practice, $\mathbf{W}_i^{(c)}, i = 1, 2, ..., l$ is implemented as $\mathbf{W}_i^{(c)} = \mathbf{W}_i'(\mathbf{W}_i')^\top$, as a metric learning approach for the generalization of Mahalanobis distance [21], in order to capture the feature correlation between nodes from two different networks. In attention-based multi-network aggregation, the output is represented as:

$$\mathbf{X}_2 = \phi(\mathbf{B}_1 \mathbf{U}_1 \mathbf{W}^{(a)} \mathbf{U}_2^\top \mathbf{B}_2^\top) \tag{6}$$

where $\mathbf{W}^{(a)}$ is the parameter matrix, $\mathbf{B}_1$ and $\mathbf{B}_2$ are attention score matrices for $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. For instance, the attention score of node $(i, j) \in \mathcal{G}_1$ is calculated as:

$$\mathbf{B}_1(i, j) = \frac{\exp(< \mathbf{u}_i, \mathbf{u}_j >)}{\sum_{k=1}^{n_1} \exp(< \mathbf{u}_i, \mathbf{u}_k >)} \tag{7}$$

In order to instantiate $b_{\boldsymbol{\Theta}}(\mathbf{F}_1, \mathbf{F}_2, \mathbf{H})$), similar to the first term $a_{\mathbf{W}}(\mathbf{F}_1, \mathbf{F}_2, \tilde{\mathbf{A}}_1, \tilde{\mathbf{A}}_2)$, we can also adopt two types of parallel aggregation layers. The first one is the direct neural aggregation from prior multi-network association, and the second one is via attention schema. However, since the entries of the prior multi-network association matrix $\mathbf{H}$ is often real values or multi-class categorical rates, it is unreasonable to directly use $\mathbf{H}$ for cross-network feature aggregation. Thus the prior multi-network association-based multi-network aggregation is only adopted when $\mathbf{H}$ contains binary associations. The two types of multi-network aggregation modules are shown as follows.

$$\mathbf{X}_3 = \phi(\mathbf{U}_1 \mathbf{H} \mathbf{U}_2^\top), \ \mathbf{X}_4 = \phi(\mathbf{U}_1 \mathbf{C} \mathbf{U}_2^\top) \tag{8}$$

where the cross-network attention score matrix $\mathbf{C}$ is calculated as $\mathbf{C}(i, j) = \frac{\exp(<\mathbf{u}_i, \mathbf{u}_j>)}{\sum_{k=1}^{n_2} \exp(<\mathbf{u}_i, \mathbf{u}_k>)}$ for $i \in \mathcal{G}_1, j \in \mathcal{G}_2$.

Putting everything together, as shown in Figure 1, the intermediate multi-network association matrices $\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4$ consist of the hidden representation tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times 4}$ for the multi-network association solution. We apply a fully connected layer on $\boldsymbol{\mathcal{X}}$ for obtaining the final multi-network association $\mathbf{X} = \text{bmm}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}})$ where $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{n_1 \times 4 \times 1}$ is the parameter tensor, and bmm() is the batch matrix multiplication operation.

3.3 Low-rank Model

The idea of low-rank model is to generate the embeddings for nodes of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively, instead of conducting bi-linear neural aggregation. Similar to the base model, we consider both the direct neural aggregation from the original network topology, and the neural aggregation from the within network attentions. First, given two networks $\mathcal{G}_1 = \{\mathbf{A}_1, \mathbf{F}_1\}, \mathcal{G}_2 = \{\mathbf{A}_2, \mathbf{F}_2\}$ with n_1, n_2 nodes respectively, the node features are fed into an MLP layer for obtaining the hidden features $\mathbf{U}_1^{(h)} = \text{MLP}_1^{(l)}(\mathbf{F}_1)$ and $\mathbf{U}_2^{(h)} = \text{MLP}_2^{(l)}(\mathbf{F}_2)$. $\mathbf{U}_1^{(h)}, \mathbf{U}_2^{(h)}$ are then fed into two parallel CGE-based and attention-based neural modules for generating the hidden representations of the node features for two networks separately. We take $\mathbf{U}_1^{(h)}$ as an example, and the process for $\mathbf{U}_2^{(h)}$ is similar. The updated node hidden representations after an l-layer CGE module is:

$$\mathbf{U}_1^{(l+1)} = \phi((\text{diag}(\boldsymbol{\sigma}) + (\mathbf{I} - \text{diag}(\boldsymbol{\sigma}))\tilde{\mathbf{A}})\mathbf{U}_1^{(l)} \mathbf{W}^{(l)}) \tag{9}$$

where $\mathbf{U}_1^{(0)} = \mathbf{U}_1^{(h)}$, $\boldsymbol{\Theta}^{(l)}$ is the parameters for the l-th layer, and $\phi()$ is an activation function. After L layers, we obtain $\mathbf{U}_1^{(L)}$. The updated node hidden representations after the attention-based neural aggregation module is:

$$\mathbf{U}_1^{(att)} = \phi(\mathbf{B}_1 \mathbf{U}_1^{(h)} \mathbf{W}^{(att)}) \tag{10}$$

where the attention score matrix $\mathbf{B}_1$ can be calculated via Eq. (7). $b_{\boldsymbol{\Theta}}(\mathbf{F}_1, \mathbf{F}_2, \mathbf{H})$ is also instantiated for $\mathcal{G}_1$ and $\mathcal{G}_2$ separately. Here, we can adopt a similar prior multi-network association-based neural aggregation when the prior $\mathbf{H}$ denotes binary or multi-class relations. For $\mathbf{H}$ with entries of K classes, we apply K neural networks, in which each neural network aggregates one class of nodes.

$$\mathbf{V}_i^{(p)} = \phi(\mathbf{H}_i \mathbf{U}_1^{(h)} \boldsymbol{\Theta}_i^{(p)}), \ \mathbf{U}_1^{(c)} = \phi(\mathbf{C}\mathbf{U}_1^{(h)} \boldsymbol{\Theta}^{(c)}) \tag{11}$$

where $\mathbf{H}_i$ is the prior multi-network association which only contains the entries of the i-th class. $\boldsymbol{\Theta}_i^{(p)}$ and $\boldsymbol{\Theta}^{(c)}$ are learnable parameters. The $\mathbf{V}_i^{(p)}$ for all classes are then concatenated and fed into an MLP for the node representation $\mathbf{U}_1^{(p)} = \text{MLP}([\mathbf{V}_1^{(p)} || \cdots || \mathbf{V}_K^{(p)}])$. The cross-network attention matrix $\mathbf{C}$ is calculated by the same method as in Eq. (8). Putting everything together, we now have four representation matrices for each network: $\mathbf{U}_1^{(L)}, \mathbf{U}_1^{(att)}, \mathbf{U}_1^{(p)}, \mathbf{U}_1^{(c)}$. We can adopt another fully connected layer to obtain a final representation $\mathbf{U}_1$[1]. The predicted multi-network association between two nodes is calculated by the dot product of the row vectors of the resulting representation matrices $\mathbf{U}_1$ and $\mathbf{U}_2$.

3.4 Training

For matrix completion, we adopt the Mean Squared Error (MSE) loss for both instantiations:

$$\mathcal{L}_1 = ||\mathbf{H} - \mathbf{M} \odot \mathbf{X}||_F^2, \ \mathcal{L}_2 = ||\mathbf{H} - \mathbf{M} \odot (\mathbf{U}_1 \mathbf{U}_2^\top)||_F^2 \tag{12}$$

where the $\mathbf{M}$ matrix is a mask of 0, 1, with 1 indicating the position of the observed prior multi-network associations. For the low-rank instantiation, the dot product of the node representations are used as the final solutions. For the regularization of the model parameters,

[1]We find that by simply adding them with the original node hidden representations, we can already achieve superior performance.

we adopt the weight decay method with 0.01 as decay factor as we find that it shows slightly better performance over L_2 regularization.

3.5 Complexity Analysis

For notation simplicity, assume that the two input networks contain n nodes and m edges. Suppose the feature dimension is d, the number of observed rating is m' and the dimension of node representations is $r < d$. For the base model, the major computation lies in the within-network and cross-network attention calculation as well as the aggregation. From Eq. (7), the within-network attention aggregation costs $O(n^2d)$. From Eq. (8), the cross-network attention aggregation costs $O(n^2d)$. The overall time complexity for the base model is $O(\#iter \cdot (n^2d))$, where $\#iter$ is the total number of iterations. The space complexity is $O(n^2)$ because of the main storage of attention score matrices. Similarly, for the low-rank model, if we do not apply the within-network and cross-network attention-based neural aggregation, the time and space complexity would be reduced to $O(L(md+nd^2)+m'd)$, and $O(m+n(d^2+r^2))$ respectively.

4 EXPERIMENTS

A - Datasets and Pre-processing. We use five widely used benchmark datasets for evaluation, namely Douban, Flixster, YahooMusic, ML-100K, and ML-1M. For the benchmark datasets, Flixster has both user-user and item-item interaction networks. Douban only contains a user-user interaction network and YahooMusic only contains an item-item interaction network. For these two datasets, we use the identity matrix as the adjacency matrix for the missing networks. For ML-100K, ML-1M, we construct their user-user and item-item interaction networks by adopting a k-nearest neighbors search via their features, and k is treated as a hyperparameter in our model. All the datasets include multi-class categorical ratings.

B - Baseline Methods. We use five baselines in our comparison, including the traditional Sylvester equation *Sylv.* [18], and recent neural network-based and GNN-based methods: *IGMC* [22], GC-MC [3], PinSage [20], and sRGCNN [16].

C - Experimental and Hyperparameter Settings. For the effectiveness comparison, we tune the hyperparameters of the model based on the best performance on the validation set. We use 2-layer GCE and attention aggregation in both instantiations on all datasets except for ML-100K and ML-1M. On these two datasets, the base model uses 3-layer GCE and attention aggregation. For the k-NN method used for generating social networks and item-item interaction networks on ML-100K and ML-1M datasets, we use $k = 10$ for the low-rank model and $k = 12$ for the base model. Further studies of the sensitivity of k will be discussed in the ablation study. The metric is the widely adopted rooted mean squared error (RMSE).

4.1 Effectiveness Results

The comparison results are shown in Table 1. The results are reported based on the average of five runs. The best performances are shown in bold fonts and the second best performances are shown with underlines. As we can see, the traditional Sylvester equation can not achieve competitive results compared to other neural network/GNN-based baseline methods, which is consistent with our discussion in Section 2. The Sylvester equation can not effectively incorporate node features, and also can not capture nonlinear relations between the observed multi-network association and the solution. Among all the neural network-based methods, the proposed framework with low-rank instantiation outperforms the rest of the baselines on Douban, Flixster, YahooMusic, ML-100K, and ML-1M datasets. Flixster (U) represents the dataset with only the usage of user-user interaction network. The performance of the proposed method slightly drops, and it shows the importance of both interaction networks of users and items in our model. Particularly, on YahooMusic dataset, the proposed method achieves 7.65% improvement over the best baseline. Among all baselines, *GC-MC* has close performance compared with our methods. *GC-MC* contains the graph encoder and the bi-linear decoder architecture which has similar effects as our proposed GNN-based neural aggregation model. This is consistent with our intuition of the effectiveness of the cross-network feature aggregation.

Table 1: RMSE comparison for geometric matrix completion.

Method	Douban	Flixster	Flixster (U)	Yahoo	ML-100K	ML-1M
Sylv.	1.220	1.244	1.276	29.403	1.403	1.323
IGMC	0.729	0.895	**0.895**	19.292	0.922	0.857
GC-MC	0.734	0.917	0.941	20.501	0.905	0.854
PinSage	0.739	0.954	0.951	22.954	0.942	0.906
sRGCNN	0.801	0.926	1.179	22.415	0.931	0.865
Ours (base)	0.762	0.911	0.934	19.277	0.915	0.851
Ours (LR)	**0.725**	**0.891**	0.916	**17.815**	**0.899**	**0.843**

4.2 Ablation Study

The ablation study results are shown in Table 2. The 'Base model (G)' and 'Base model (A)' represent the model with only GCE-based neural aggregation, and the model with only attention-based neural aggregation. The low-rank model uses the same abbreviation. The values inside the parentheses denote the maximum allocated GPU memory in one epoch, in which we use the same batch size (i.e. 50) for comparison. As we can see, firstly the proposed model outperforms all variants with both base and low-rank instantiations. Secondly, the model without the attention neural aggregation overall consumes the least GPU memory during training. On average, with only 1.24% performance drop, the models without attention neural aggregation show 16.98% less memory consumption. Furthermore, comparing with other baselines' performance in Table 1, the variant low-rank model in Table 2 still outperforms all baselines.

Table 2: Ablation study on ML-100K and ML-1M dataset.

Method	ML-100K	ML-1M
Base model	**0.915** (160Mb)	**0.851** (2,371Mb)
Base model (G)	0.932 (141Mb)	0.862 (2,099Mb)
Base model (A)	0.924 (148Mb)	0.861 (2,209Mb)
Low-rank	**0.899** (136Mb)	**0.843** (1,983Mb)
Low-rank (G)	0.902 (110Mb)	0.857 (1,476Mb)
Low-rank (A)	0.920 (116Mb)	0.853 (1,641Mb)

5 CONCLUSION

In this paper, we propose SyMGNN, a flexible neural framework for generalizing the traditional Sylvester equation towards an end-to-end neural model for multi-network mining. We further propose two specific instantiations of the SyMGNN framework for geometric matrix completion task. The experimental results show that the proposed models overall outperform baselines on all existing benchmark datasets. Furthermore, the proposed low-rank instantiation could reduce the memory consumption by 16.98% on average.

6 ACKNOWLEDGEMENT

This work is partially supported by DARPA (HR001121C0165), DHS (17STQAC00001-07-00), NIFA (2020-67021-32799), NSF (1947135, 2134079, 1939725, 2316233, and 2324770), and ARO (W911NF2110088).

REFERENCES

[1] Amrudin Agovic, Arindam Banerjee, and Snigdhansu Chatterjee. 2011. Probabilistic matrix addition. In *ICML*.

[2] Peter Benner. 2004. Factorized solution Of Sylvester equations with applications in Control. *sign (H)* 1 (2004), 2.

[3] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[4] Gang Chen, Yangqiu Song, Fei Wang, and Changshui Zhang. 2008. Semi-supervised multi-label learning by solving a sylvester equation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 410–419.

[5] Xiaokai Chu, Xinxin Fan, Di Yao, Zhihua Zhu, Jianhui Huang, and Jingping Bi. 2019. Cross-network embedding for multi-network alignment. In *The world wide web conference*. 273–284.

[6] Boxin Du, Lihui Liu, and Hanghang Tong. 2021. Sylvester Tensor Equation for Multi-Way Association. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 311–321.

[7] Boxin Du and Hanghang Tong. 2018. Fasten: Fast sylvester equation solver for graph mining. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1339–1347.

[8] Boxin Du, Si Zhang, Nan Cao, and Hanghang Tong. 2017. First: Fast interactive attributed subgraph matching. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1447–1456.

[9] Boxin Du, Si Zhang, Yuchen Yan, and Hanghang Tong. 2021. New frontiers of multi-network mining: Recent developments and future trend. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4038–4039.

[10] Gene Golub, Stephen Nash, and Charles Van Loan. 1979. A Hessenberg-Schur method for the problem AX+ XB= C. *IEEE Trans. Automat. Control* 24, 6 (1979), 909–913.

[11] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. 2019. Graph matching networks for learning the similarity of graph structured objects. In *International Conference on Machine Learning*. PMLR, 3835–3845.

[12] Zhuliu Li, Raphael Petegrosso, Shaden Smith, David Sterling, George Karypis, and Rui Kuang. 2021. Scalable Label Propagation for Multi-relational Learning on the Tensor Product of Graphs. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[13] Zhuliu Li, Wei Zhang, R Stephanie Huang, and Rui Kuang. 2019. Learning a Low-Rank Tensor of Pharmacogenomic Multi-relations from Biomedical Networks. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 409–418.

[14] Hanxiao Liu and Yiming Yang. 2016. Cross-graph learning of multi-relational associations. In *International Conference on Machine Learning*. PMLR, 2235–2243.

[15] Lihui Liu, Boxin Du, Hanghang Tong, et al. 2019. G-finder: Approximate attributed subgraph matching. In *2019 IEEE international conference on big data (big data)*. IEEE, 513–522.

[16] Federico Monti, Michael M Bronstein, and Xavier Bresson. 2017. Geometric matrix completion with recurrent multi-graph neural networks. *arXiv preprint arXiv:1704.06803* (2017).

[17] Jiliang Tang, Xia Hu, and Huan Liu. 2013. Social recommendation: a review. *Social Network Analysis and Mining* 3, 4 (2013), 1113–1133.

[18] Eugene L Wachspress. 1988. Iterative solution of the Lyapunov matrix equation. *Applied Mathematics Letters* 1, 1 (1988), 87–90.

[19] Kai-Lang Yao, Wu-Jun Li, Jianbo Yang, and Xinyan Lu. 2018. Convolutional geometric matrix completion. *arXiv preprint arXiv:1803.00754* (2018).

[20] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[21] Tomoki Yoshida, Ichiro Takeuchi, and Masayuki Karasuyama. 2021. Distance metric learning for graph structured data. *Machine Learning* (2021), 1–47.

[22] Muhan Zhang and Yixin Chen. 2019. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058* (2019).

[23] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1345–1354.

[24] Si Zhang and Hanghang Tong. 2018. Attributed network alignment: Problem definitions and fast solutions. *IEEE Transactions on Knowledge and Data Engineering* 31, 9 (2018), 1680–1692.

[25] Qinghai Zhou, Liangyue Li, Nan Cao, Lei Ying, and Hanghang Tong. 2019. ADMIRING: Adversarial multi-network mining. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1522–1527.

[26] Qinghai Zhou, Liangyue Li, Nan Cao, Lei Ying, and Hanghang Tong. 2021. Adversarial Attacks on Multi-Network Mining: Problem Definition and Fast Solutions. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[27] Qinghai Zhou, Liangyue Li, Xintao Wu, Nan Cao, Lei Ying, and Hanghang Tong. 2021. Attent: Active attributed network alignment. In *Proceedings of the Web Conference 2021*. 3896–3906.