ATTENTION-BASED GRAPH CONVOLUTIONAL NETWORK FOR RECOMMENDATION SYSTEM

Chenyuan Feng, Zuozhu Liu, Shaowei Lin and Tony Q.S. Quek

Singapore University of Technology and Design, Singapore

ABSTRACT

Matrix completion with rating data and auxiliary information for users and items is a challenging task in recommendation systems. In this paper, we propose an end-to-end architecture named Attention-based Graph Convolutional Network (AGCN) to embed both rating data and auxiliary information in a unified space, and subsequently learn low-rank dense representations via graph convolutional networks and attention layers. Compared to previous work, AGCN reduces computational complexity with Chebyshev polynomial graph filters. The introduced attention layer, which encourages weighing the neighbor information to learn more expressive structural graph representations, can improve the prediction accuracy, and lead to faster and more stable convergence. Experimental results show that our model can perform better and converge faster than current state-of-the-art methods on the real-world MovieLens and Flixster datasets.

Index Terms— Graph Signal Processing, Attention, Graph Convolutional Network, Recommendation System

1. INTRODUCTION

Matrix Factorization (MF) is the one of the most often-used techniques for recommendation systems by approximating the observed sparse rating matrix with two low-rank dense matrices that correspond to latent features for users and items, respectively, as shown in Fig.1. However, besides rating matrix, side information for users and items, e.g., relationship among users from social networks, can also be involved to further explore behavior patterns, leading to more precise recommendations [1–4]. Recent works attempt to directly capture structural information among users/items/ratings by constructing the rating matrix and side information as graphs [5, 6].

Graph convolutional networks (GCNs) are proposed to extend convolutional neural network to directly operate on the spectral or spatial domain of graphs for effective representation learning [7–9]. Subsequent works applying GCNs for recommendation systems demonstrate its superior performance over traditional MF-based methods. However, they are usually shallow architectures and still rely on additional computational units such as recurrent networks [5–7], leaving the design of lightweight and deeper frameworks as future work. Moreover, the propagation layers of existing GCNs adopt a static and non-adaptive mechanism, which fails to capture the relevance between different neighbors.

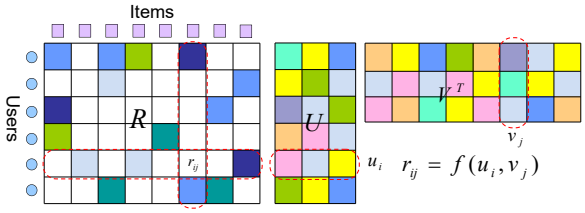


Fig. 1. The MF representations for score-based recommendation systems. The rows of $\mathbf{U}$ and the columns of $\mathbf{V}^{\mathrm{T}}$ can be viewed as latent feature representations of users and items, their inner product provides an prediction for ratings.

With a long history in neuroscience, attention mechanism [10] has been used in plenty of machine learning models and tasks lately to improve performance [11–13]. Essentially, attention is a technique that learns a distribution over the available input and then zooms in or out on it accordingly in order to reach a more reliable decision making process. In graph signal processing fields, attention can be applied to learn the relevance among nodes to facilitate information propagation and learn structural features for graphs [14, 15] .

Motivated by these issues and advancements, in this paper, we propose a novel attention-based graph convolutional network (AGCN) model which learns expressive user and item features for matrix completion. In AGCN, both rating data and complementary side information are embedded to the same space and subsequently processed via polynomial Chebyshev graph filters to deal with heterogeneous information sources [8]. To better explore the underlying graph structure, especially for extremely sparse graphs, we introduce an attention layer over GCNs to emphasize on important neighbors for information update. Compared to previous GCN-based work, AGCN is applicable for much deeper architectures without complicated neural network frameworks. Experimental results on two real-world datasets show that AGCN can achieve better performance and converge much faster and more stable than state-of-the-art models.

2. PROBLEM FORMULATION

2.1. Regularized Matrix Factorization

Given a collective interaction rating matrix $\mathbf{R} \in \mathbb{R}^{N_u \times N_v}$, where N_u and N_v denote the number of users and items respectively, the element r_{ij} denotes the ratings given by user i to item j. The observed elements constitute a small subset Ω of the rating matrix $\mathbf{R}$. The side information, such as user demographic information and item attributes or other content information, can be encoded as raw feature vectors $\mathbf{p}_i \in \mathbb{R}^P$ for user i and $\mathbf{q}_j \in \mathbb{R}^Q$ for item j. MF methods aim to learn latent feature representations for users and items and reconstruct the whole matrix by the product of corresponding feature matrices, namely $\mathbf{R} \approx \mathbf{U}\mathbf{V}^T$ where $\mathbf{U} \in \mathbb{R}^{N_u \times K}$ and $\mathbf{V} \in \mathbb{R}^{N_v \times K}$ with $K \ll N_u, N_v$ denoting the dimension of latent feature vector.

To tackle with this problem, a well-posed method is low rank assumption with a tight convex relaxation [16], which is equivalent to minimizing the following loss function:

$$\mathcal{J}(\boldsymbol{U}, \boldsymbol{V}) = \left\| \Omega \circ (\mathbf{R} - \mathbf{U}\mathbf{V}^T) \right\|_F^2 + \lambda \left(\|\boldsymbol{U}\|_\star^2 + \|\boldsymbol{V}\|_\star^2 \right), \quad (1)$$

where the first term is the reconstruction error for observed data with $\Omega \circ (\cdot)$ is the binary projection operator that only counts observed entries of the matrix which lie in the set Ω, and $\|\cdot\|_F$ denotes the Frobenius norm. And the second term is the regularization term with different nuclear norm $\|\cdot\|_\star^2$ for different model [16], and λ denotes the regularization factor.

2.2. Graph Embedding

How to embed recommendation systems as graphs varies for different models. The graphs mainly fall into two categories: user-item bipartite graph or user-item multi-graph as shown in Fig.2. For user-item bipartite graph, users and items are denoted as nodes while entries in rating matrix are considered as edges [8] [17] [6]. Since there are no edges between any user node pairs or item node pairs, the recommendation is of bipartite structure and the reconstruction task can be cast as a link prediction problem.

In contrast, geometric structure within matrix rows and columns exist in user-item multi-graph, resulting from complementary graphs, e.g., graphs from users' social network data or item relationship data [18] [5] [16]. The side information can be encoded as undirected weighted row graph $\mathcal{G}_r = (\{u_1, ..., u_{N_u}\}, \mathcal{E}_r, \mathcal{W}_r)$ and column graph $\mathcal{G}_c = (\{v_1, ..., v_{N_v}\}, \mathcal{E}_c, \mathcal{W}_c)$. Take $\mathcal{G}_r$ as an example, $\{u_1, ..., u_{N_u}\}$ is the set of user nodes, $\mathcal{E}_u$ is the set of edges, and each entry $w_{ij}^r = \text{sim}(\mathbf{p}_i, \mathbf{p}_j)$ measures the similarity between node u_i and u_j, where $\text{sim}(\cdot)$ is a function to compute the similarity. In our experiments, this row graph comes from the demographic information for users. The column graph $\mathcal{G}_r$ for items is defined in the same fashion.

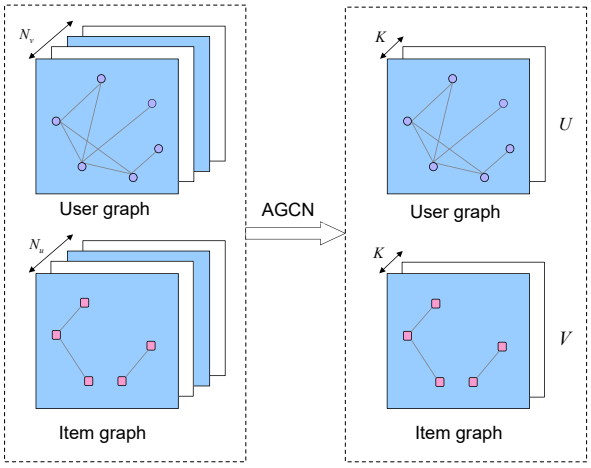


Fig. 2. Example of user-item multi-graph. Take user graph as example, $N_u \times N_v$-dimension rating matrix can be embedded as N_u user nodes and each user node associates with N_v-dimension sparse graph signal as its rating behavior. Auxiliary content information is used to build edges among user nodes. A K-dimension dense feature vector is learned for each user node via AGCN. The column graph is defined in similar manner.

The rating data is represented as vectorized graph signals for each node, i.e., the graph signal $\mathbf{r}_{i,:}$ associated with user node u_i is i-th row of rating matrix $\mathbf{R}$, while for the item node v_j is j-th column $\mathbf{r}_{:,j}$. Matrix factorization problem following this graph embedding mechanism can be cast as mapping high-dimensional sparse graph signals to low-dimensional dense features.

2.3. Objective Function

In graph signal processing (GSP), the normalized graph Laplacian is defined as a symmetric semi-definite matrix $\mathcal{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathcal{W}\mathbf{D}^{-1/2}$, where $\mathbf{D} = \text{diag}\left(\sum_{i \neq j} w_{ij}\right)$ is the degree matrix and $\mathbf{I}$ is the identity matrix. The Laplacian matrices for row and column graph are denoted as $\mathcal{L}_\text{r}$ and $\mathcal{L}_\text{c}$. Since $\mathcal{L}$ is symmetric, it can be decomposed as $\mathcal{L} = \Phi\Lambda\Phi^T$ and has a complete set of orthonormal eigenvectors, denoted as $\Phi = (\phi_1, \phi_2, ..., \phi_n)$, and the diagonal matrix of corresponding eigenvalues $\Lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$. The graph smoothness is defined as the graph total variation [19] of all graph signals, i.e., the smoothness is $\|\mathbf{U}\|_{\mathcal{G}_r}^2 = \text{trace}(\mathbf{U}^T\mathcal{L}_r\mathbf{U})$ for row graph and $\|\mathbf{V}\|_{\mathcal{G}_c}^2 = \text{trace}(\mathbf{V}\mathcal{L}_c\mathbf{V}^T)$ for column graph. Under these settings, the loss function in (1) can be rewritten as

$$\mathcal{J}(\mathbf{U}, \mathbf{V}) = \left\| \Omega \circ (\mathbf{R} - \mathbf{U}\mathbf{V}^T) \right\|_F^2 + \lambda \left(\|\mathbf{U}\|_{\mathcal{G}_r}^2 + \|\mathbf{V}\|_{\mathcal{G}_c}^2 \right) \quad (2)$$

3. METHODOLOGY

3.1. Graph Convolution

The AGCN model consists of two main components: the GCN block and the attention layer. It takes as input the user-item multi-graph and outputs two low-rank dense matrices $\mathbf{U}, \mathbf{V}$ for completion of $\mathbf{R}$, as shown in Fig.3 (a).

Here we mathematically define the convolution operations for a M-layer GCN. Considering a normalized polynomial filters with Chebyshev coefficient proposed in [9, 20], the convolution at the m-th layer, which receives input as $\tilde{\mathbf{H}}^{(m)}$ and outputs $\mathbf{H}^{(m+1)}$, is defined as

$$\mathbf{H}^{(m+1)} = \sigma(\tilde{\mathbf{H}}^{(m)}\mathbf{W}_{\theta}^{(m)}(\mathcal{L})), \tag{3}$$

where σ is the activation function (ReLU), and $\mathbf{W}_{\theta}^{(m)}(\mathcal{L}) = \left\{w_{\theta_{ij}}^{(m)}\right\}$ is the weight matrix defined as

$$w_{\theta_{ij}}^{(m)} = \sum_{p=0}^{P-1} \theta_{ij,p}^{(m)} \mathbf{T}_p(\hat{\mathcal{L}}), \tag{4}$$

where $\theta_{ij,p}$ is the polynomial filter coefficients to be learned, $\hat{\mathcal{L}} = 2\mathcal{L}/\lambda_{\max} - \mathbf{I}$ whose eigenvalues are in the interval $[-1, 1]$, and $\mathbf{T}_p(\cdot)$ is the recursively generated Chebyshev polynomial. P denotes the order of the polynomial, which means the Laplacian is a local operator with P-hop spatial neighborhood. For row graph, $\tilde{\mathbf{H}}^{(0)} = \mathbf{R}$ and Laplacian matrix $\mathcal{L} = \mathcal{L}_r$, for column graph, $\tilde{\mathbf{H}}^{(0)} = \mathbf{R}^T$ and $\mathcal{L} = \mathcal{L}_c$.

3.2. Attention Mechanism

Each graph convolutional layer is followed by an attention layer defined as

$$\tilde{\mathbf{H}}^{(m)} = \mathbf{A}^{(m)}\mathbf{H}^{(m)} \tag{5}$$

where the attention matrix $\mathbf{A}^{(m)}$ is also a function of the input signal $\mathbf{H}^{(m)}$. More specifically, the output weighted feature vector of node j is

$$\tilde{\mathbf{h}}_j^{(m)} = \sum_{i \in \mathcal{N}_j} \alpha_{ij}^{(m)} \mathbf{h}_i^{(m)} \tag{6}$$

where $\mathcal{N}_j$ denotes the neighborhood of node j including itself, and the attention coefficient is computed as

$$\alpha_{ij}^{(m)} = \frac{\exp\left(\beta^{(m)} \cos\left(\mathbf{h}_i^{(m)}, \mathbf{h}_j^{(m)}\right)\right)}{\sum_{i' \in \mathcal{N}_j} \exp\left(\beta^{(m)} \cos\left(\mathbf{h}_{i'}^{(m)}, \mathbf{h}_j^{(m)}\right)\right)} \tag{7}$$

with $\cos(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a}^{\mathrm{T}}\mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|}$, and $\beta^{(m)}$ is the scalar parameters for the attention layer. Essentially, the attention score $\alpha_{ij}^{(m)}$ captures how relevant node i is to node j. By putting more attention on important neighbors, the node representations can be updated more efficiently and hence leading to faster convergence.

3.3. Model Justification

Convergence. The AGCN model can be interpreted as a graph-based version of the Weisfeiler-Lehman algorithm [21]. For all nodes $i \in \mathcal{G}$, the 1-D Weisfeiler-Lehman algorithm first gets node features $\{x_j\}$ of neighboring nodes $\{j \in \mathcal{N}_i\}$, and then updates them as $x_i \leftarrow \text{hash}(\sum_j x_j)$, where hash$(\cdot)$ is an injective hash function. Repeat multiple steps or until convergence. In our AGCN model, the layer-wise propagation rule can be interpreted as a differentiable and parameterized variant of the hash function as $\mathbf{h}_j^{(m+1)} = \sigma(\sum_{i \in \mathcal{N}_j} \alpha_{ij}^{(m)} \mathbf{W}^{(m)} \mathbf{h}_i^{(m)})$. This update rule becomes stable in practice when choosing an appropriate non-linear activation function (e.g., ReLU) and initializing the random weight matrix orthogonally [22]. The pure GCN model is a special case of our model when attention score $\alpha_{ij}^{(t)} = (|\mathcal{N}_i|\,|\mathcal{N}_j|)^{-1/2}$.

Model complexity. The proposed attention mechanism over neighbors in (7) learns which neighbors are more relevant and weighs their contributions accordingly. Similar to [13,23], we adopt a very simple attention formulation with only one parameter for each layer. It is critical to reduce model complexity for successfully train in highly sparse datasets. Complex attention mechanisms, such as self-attention [15] or multi-head attention [14], are more suitable for rich information datasets task but not recommendation system, as they might resulting in unstable training and lower accuracy.

4. EXPERIMENT

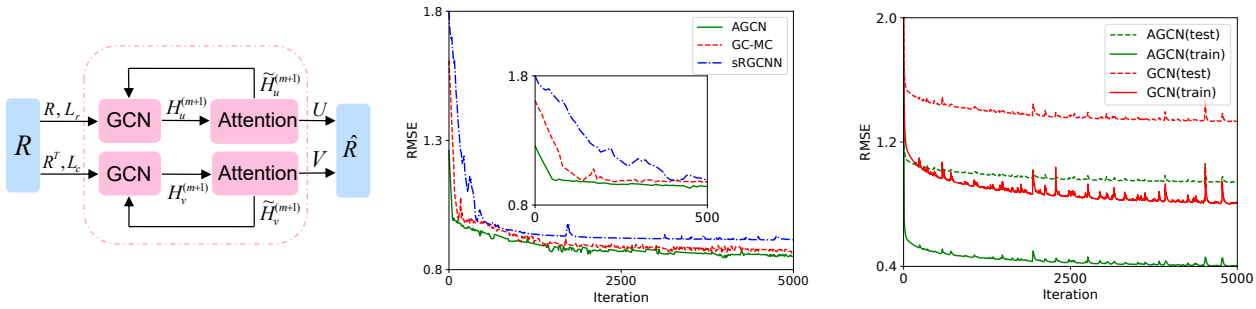
4.1. Experimental Setup

We evaluate our model on two standard movie recommendation benchmark datasets: the MovieLens-100K and Flixster datasets [5]. The statistics of the sparse rating matrices are shown in Table.1. The side information for MovieLens-100K is user demographics (age, gender and occupation) and item genres. Flixster provides relationship information among users and among items as well. We randomly split the datasets with 70% for training, 20% for validation and 10% for test.

We stack $M = 6$ graph convolution and attention layers. The dimension of latent feature vector is set to $K = 32$ and the order of Chebyshev polynomials is $P = 5$. Adam [24] is used for optimization with learning rate $\eta = 0.001$. Through cross-validation, we set the regularization coefficient as $\lambda = 0.001$. All models are implemented with TensorFlow [25].

4.2. Result Analysis

Performance Evaluation. We compare AGCN with state-of-the-art methods including GRALS [18], AFM [15], sRGCNN [5] and GC-MC [6]. The root-mean-square errors (RMSE) are reported in Table 2. First of all, we can observe that AGCN outperforms all baselines on both datasets. The



(a) (b) (c)

Fig. 3. (a). AGCN architecture for matrix factorization problem. We have two co-trained GCN+Attention blocks for users and items, respectively. (b). The RMSE of AGCN, GC-MC and sRGCNN during training. We zoom in on the first 500 iterations to justify the convergence speed. (d). Comparison of RMSE for AGCN and GCNs (without attention).

Table 1. Statistics of the rating matrices.

Dataset	#user	#item	#rating	sparsity
MovieLens-100K	943	1682	100,000	6.3%
Flixster	3000	3000	26173	0.29%

Table 2. RMSE on MovieLens-100K and Flixster datasets.

Method	MovieLens-100K	Flixster
GRALS([18])	0.945	1.245
sRGCNN([5])	0.929	0.926
AFM([15])	0.913	0.922
GC-MC([6])	0.905	0.917
AGCN(ours)	**0.898**	**0.901**

RMSE of AGCN is much lower than traditional graph-based methods, i.e., it is 4.7% and 34.4% lower than GRALS for MovieLens-100K and Flixster, respectively. Compared to GCN-based methods, AGCN reduces the RMSE by 0.7%-3.1% and 0.6%-2.5% for the two tasks. We speculate that the introduced attention layer could assist GCNs to concentrate more on important node information, enabling learning more expressive representations. This is demonstrated in Fig.3.(c), where AGCN converges to a much lower RMSE than GCN-based models. It is also, to some extent, consistent with the observed lower RMSE achieved by AFM than by GRALS, where both of them are graph-based methods without GCNs while AFM employs additional attention mechanism.

Convergence. Here we justify the convergence speed and stability of our model. As Fig.3.(b) shows, besides the lower RMSE, AGCN also converges much faster than GC-MC and sRGCNN, especially in the first hundreds of iterations. We shall attribute this observation to the attention layers, which always motivate the GCNs to update the node representations in a right manner. We also show how the attention layer can help stabilize the convergence. As shown in Fig.3(c), the RMSE experiences much less oscillations and hence attains a lower variance by adding attention layers to the GCNs.

Table 3. Computation and model complexity.

Method	Computation	Model
GRALS([18])	$O(\mathcal{N}_u + \mathcal{N}_v)$	$O(\mathcal{N}_u + \mathcal{N}_v)$
sRGCNN([5])	$O(\mathcal{N}_u + \mathcal{N}_v)$	$O(1)$
AFM([15])	$O(\mathcal{N}_u\mathcal{N}_v)$	$O(\mathcal{N}_u\mathcal{N}_v)$
GC-MC([6])	$O(\mathcal{N}_u + \mathcal{N}_v)$	$O(\mathcal{N}_u + \mathcal{N}_v)$
AGCN(ours)	$O(\mathcal{N}_u + \mathcal{N}_v)$	$O(\mathcal{N}_u + \mathcal{N}_v)$

Computational Efficiency. Another key advantage of of our model is the low computation and model complexity. As Table 2 shows, AFM scales quadratically as $O(\mathcal{N}_u\mathcal{N}_v)$, while the rest including our model scale linearly as $O(\mathcal{N}_u + \mathcal{N}_v)$. Moreover, the model parameters and memory storage required by attention in AFM are both quadratic to the number of non-zero features, i.e., $(O((\text{nnz}(R))^2))$ where $nnz(\cdot)$ counts the non-zero entries. In contrast, our attention strategy only introduces $K^2 \ll (nnz(R))^2$ for memory storage and one scalar parameter to learn in each layer. Along with the better performance from AGCN, we can conclude that our model is both simple and effective.

5. CONCLUSION

We propose an attention-based graph neural network which integrates GCNs and attention mechanisms. The proposed AGCN can learn low-dimension discriminative user/item latent representations and perform better than current state-of-the-art methods in standard datasets. Compared with other GCN models, the improvement is due to the utilization of additional attention layers. Compared with attention-based collaborative filtering models, our attention layer is much simpler but still effective enough. Future work includes designing more powerful GCNs along with attention mechanisms to improve prediction accuracy and computational efficiency.

6. REFERENCES

[1] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonengative matrix factorization for data representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

[2] H. Ma and et al, "Recommender systems with social regularization," in *Proceedings of the 2011 ACM International Conference on Web Search and Data Mining (WSDM)*, Feb. 2011, pp. 287–296.

[3] F. Monti, M. M. Bronstein, and X. Bresson, "Deep geometric matrix completion: A new way for recommender systems," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6852–6856.

[4] W. Huang, A. G. Marques, and A. R. Ribeiro, "Rating prediction via graph signal processing," *IEEE Transactions on Signal Processing*, vol. 66, pp. 50665081, 2018.

[5] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Conference on Neural Information Processing Systems (NIPS)*, 2017, pp. 3700–3709.

[6] R. V. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *CoRR*, vol. abs/1706.02263, 2017.

[7] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *International Conference on Learning Representations (ICLR)*, 2014.

[8] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[11] A. Vaswani and et al, "Attention is all you need," in *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[12] J. Chen and et al, "Attentive collaborative filtering: Multimedia recommendation with feature- and item-level attention," *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 335–344, 2017.

[13] K. K. Thekumparampil, C. Wang, S. Oh, and L. Li, "Attention-based graph neural network for semi-supervised learning," in *International Conference on Learning Representations (ICLR)*, 2018.

[14] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations (ICLR)*, 2018.

[15] J. Xiao and et al, "Attentional factorization machines: learning the weight of feature interactions via attention networks," *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3119–3125, 2017.

[16] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[17] X. Li and H. Chen, "Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach," *Decision Support Systems*, vol. 2, no. 52, pp. 880–890, 2013.

[18] N. Rao, H. Yu, P. Ravikumar, and I. S. Dhillon, "Collaborative filtering with graph information: Consistency and scalable methods," *Conference on Neural Information Processing Systems (NIPS)*, pp. 2098–2106, Dec 2015.

[19] A. Sandryhalia and J. M. F. Moura, "Big data analysis with signal processing on graphs," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[21] S. Nino and et al, "Weisfeiler-lehman graph kernels," *The Journal of Machine Learning Research (JMLR)*, vol. 12, pp. 2539–2561, Nov. 2011.

[22] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, vol. PMLR, no. 9, pp. 249–256, 2010.

[23] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *CoRR*, vol. abs/1410.5401, 2014.

[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.

[25] M. Abadi and et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," in *arXiv preprint arXiv:1603.04467*, 2016.