Recommendation System using Geometric Matrix Completion with Multi-Graph Neural Networks

Shashank (sshashan@stanford.edu)

1. Introduction:

Recommender systems have become a central part of modern intelligent systems. Recommending movies on Netflix, friends on Facebook, furniture on Amazon, jobs on LinkedIn are a few examples of the main purpose of these systems. Recommender systems can be broadly divided into two categories based on approach collaborative and content filtering techniques. In collaborative filtering customer ratings are used to suggest new recommendations while in content based filtering similarities between products and customers is used.

A typical collaborative filtering scenario in recommendation systems can be modeled as a *matrix completion* problem. columns and rows represent users and items, respectively, and matrix values represent a score determining user affinity for item. A famous example is the Netflix challenge offered in 2009 and carrying a 1M$ prize for the algorithm that can best predict user ratings for movies based on previous ratings. This matrix can be modeled in form of column and row graph representing similarity of users and items, respectively. Recently there have been deep learning approaches to solve matrix completion problem specifically CNN. But CNN can't be applied directly on non euclidean domains suc as graphs. This strongly motivates the development of *geometric deep learning* techniques that can mathematically deal with graph-structured data.

1.1. Main contribution

In this work, we treat matrix completion problem as deep learning on graph-structured data. We introduce a novel neural network architecture that is able to extract local stationary patterns from the high-dimensional spaces of users and items, and use these meaningful representations to infer the non-linear temporal diffusion mechanism of ratings.

2. Literature review:

2.1. Matrix Completion

Recovering the missing values of a matrix given a small fraction of its entries can be considered as low rank approximation of matrix.

$$\min_{\mathbf{X}} \ \mathrm{rank}(\mathbf{X}) \quad \text{s.t.} \quad x_{ij} = y_{ij}, \ \forall ij \in \Omega, \tag{1}$$

where $\mathbf{X}$ denotes the matrix to recover, Ω is the set of the known entries and y_{ij} are their values. Above optimization problem can be posed as following equation by adding constraint in equation.

$$\min_{\mathbf{X}} \ \mathrm{rank}(\mathbf{X}) + \frac{\mu}{2}\|\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{Y})\|_{\mathrm{F}}^2, \tag{2}$$

But rank minimization is NP-hard combinatorial problem. The tightest possible convex relaxation of the previous problem is

$$\min_{\mathbf{X}} \ \|\mathbf{X}\|_{\star} + \frac{\mu}{2}\|\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{Y})\|_{\mathrm{F}}^2, \tag{3}$$

where $\|\cdot\|_{\star}$ is the nuclear norm of a matrix equal to the sum of its singular values.

2.2. Geometric matrix completion

An alternative relaxation of the rank operator in (1) is to constraint the space of solutions to be smooth w.r.t. some geometric structure of the matrix rows and columns.The simplest model is proximity structure represented as an undirected weighted column graph $\mathcal{G}_c = (\{1, \ldots, n\}, \mathcal{E}_c, \mathbf{W}_c)$ with *adjacency matrix* $\mathbf{W}_c = (w_{ij}^c)$, where $w_{ij}^c = w_{ji}^c$, $w_{ij}^c = 0$ if $(i, j) \notin \mathcal{E}_c$ and $w_{ij}^c > 0$ if $(i, j) \in \mathcal{E}_c$. Where column graph can be thought of as a social network capturing relations between users and the similarity of their tastes. The row graph $\mathcal{G}_r = (\{1, \ldots, m\}, \mathcal{E}_r, \mathbf{W}_r)$ representing the similarities of the items is defined in a similar manner.

Considering the columns (respectively, rows) of matrix $\mathbf{X}$ as vector-valued functions on the column graph $\mathcal{G}_c$ (respectively, row graph $\mathcal{G}_r$), their smoothness can be expressed as the *Dirichlet norm* $\|\mathbf{X}\|_{\mathcal{G}_r}^2 = \mathrm{trace}(\mathbf{X}^\top \mathbf{\Delta}_r \mathbf{X})$ (respecitvely, $\|\mathbf{X}\|_{\mathcal{G}_c}^2 = \mathrm{trace}(\mathbf{X}\mathbf{\Delta}_c\mathbf{X}^\top)$). The *geometric matrix completion* problem thus boils down to minimizing

$$\min_{\mathbf{X}} \ \|\mathbf{X}\|_{\mathcal{G}_r}^2 + \|\mathbf{X}\|_{\mathcal{G}_c}^2 + \frac{\mu}{2}\|\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{Y})\|_{\mathrm{F}}^2, \tag{4}$$

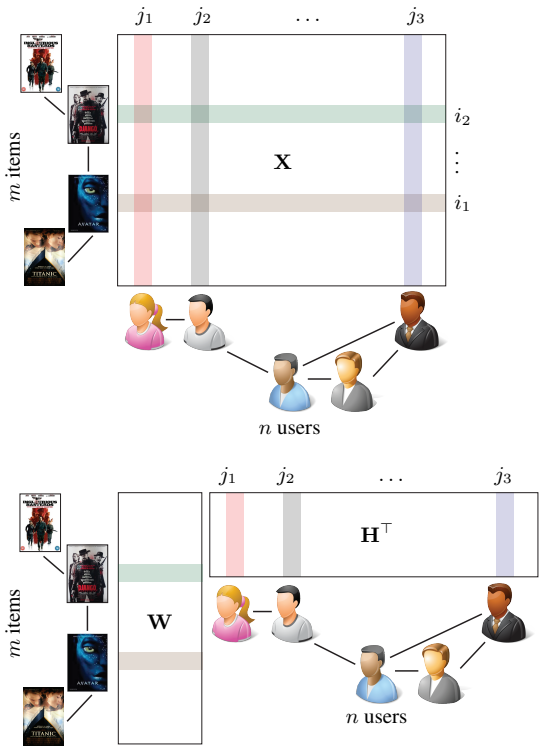

Figure 1. Full (top) and factorized (bottom) geometric matrix completion representations. The column and row graphs represent the relationships between users and items, respectively.

2.3. Factorized models

Although matrix completion algorithms introduced in the previous section are well-posed as convex optimization problems.However, the variables in this formulation are the full $m \times n$ matrix $\mathbf{X}$, makes it hard to scale up to large matrices such as the Netflix challenge.
A solution is to use a factorized representation $\mathbf{X} = \mathbf{W}\mathbf{H}^\top$, where $\mathbf{W}, \mathbf{H}$ are $m \times r$ and $n \times r$ matrices, respectively, with $r \ll \min(m, n)$. Factorized formulation of the graph-based minimization problem can be written (4) as

$$\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{W}\|^2_{\mathcal{G}_r} + \frac{1}{2}\|\mathbf{H}\|^2_{\mathcal{G}_c} + \frac{\mu}{2}\|\mathbf{\Omega} \circ (\mathbf{W}\mathbf{H}^\top - \mathbf{Y})\|^2_{\mathrm{F}}. \quad (5)$$

2.4. Deep learning on graphs

The key idea to Author work is *geometric deep learning*, an extension of the popular CNNs to graphs. A graph Laplacian admits a spectral eigendecomposition of the form $\mathbf{\Delta} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^\top$, where $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \dots \boldsymbol{\phi}_n)$ denotes the matrix of orthonormal eigenvectors and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ is the diagonal matrix of the corresponding eigenvalues. The eigenvectors play the role of Fourier atoms in classical harmonic analysis and the eigenvalues can be interpreted as frequencies.

The *spectral convolution* of two functions $\mathbf{x}, \mathbf{y}$ can be defined as the element-wise product of the respective Fourier transforms,

$$\mathbf{x} \star \mathbf{y} = \mathbf{\Phi}(\mathbf{\Phi}^\top \mathbf{x}) \circ (\mathbf{\Phi}^\top \mathbf{y}) = \mathbf{\Phi}\,\mathrm{diag}(\hat{y}_1, \dots, \hat{y}_n)\,\hat{\mathbf{x}}. \quad (6)$$

Where *graph Fourier transform* is $\hat{\mathbf{x}} = \mathbf{\Phi}^\top \mathbf{x}$. A spectral convolutional layer has the form

$$\tilde{\mathbf{x}}_l = \xi \left(\sum_{l'=1}^{q'} \mathbf{\Phi}\hat{\mathbf{Y}}_{ll'}\mathbf{\Phi}^\top \mathbf{x}_{l'} \right), \quad l = 1, \dots, q, \quad (7)$$

where q', q denote the number of input and output channels, respectively, $\hat{\mathbf{Y}}_{ll'} = \mathrm{diag}(\hat{y}_{ll',1}, \dots, \hat{y}_{ll',n})$ is a diagonal matrix of spectral multipliers representing a learnable filter in the spectral domain, and ξ is a nonlinearity (e.g. ReLU) applied on the vertex-wise function values. Since there is no efficient FFT algorithms on genral graphs above operation is costly $\mathcal{O}(n^2)$.
To address these issues, Author has proposed an explicit expansion in the Chebyshev polynomial basis to represent the spectral filters

$$\tau_{\boldsymbol{\theta}}(\tilde{\mathbf{\Delta}}) = \sum_{j=0}^{p-1} \theta_j T_j(\tilde{\mathbf{\Delta}}) = \sum_{j=0}^{p-1} \theta_j \mathbf{\Phi} T_j(\tilde{\mathbf{\Lambda}}) \mathbf{\Phi}^\top, \quad (8)$$

where $\tilde{\mathbf{\Delta}} = 2\lambda_n^{-1}\mathbf{\Delta} - \mathbf{I}$ is the rescaled Laplacian such that its eigenvalues $\tilde{\mathbf{\Lambda}} = 2\lambda_n^{-1}\mathbf{\Lambda} - \mathbf{I}$ are in the interval $[-1, 1]$, $\boldsymbol{\theta}$ is the p-dimensional vector of polynomial coefficients parametrizing the filter, and $T_j(\lambda) = 2\lambda T_{j-1}(\lambda) - T_{j-2}(\lambda)$ denotes the Chebyshev polynomial of degree j defined in a recursive manner with $T_1(\lambda) = \lambda$ and $T_0(\lambda) = 1$. [1]
Main advantage of this approach is that it does not require an explicit computation of the Laplacian eigenvectors, and due to the recursive definition of the Chebyshev polynomials, the computation of the filter incurs applying the Laplacian p times. Assuming the graph has $|\mathcal{E}| = \mathcal{O}(n)$ edges, the overall complexity is $\mathcal{O}(n)$ rather than $\mathcal{O}(n^2)$ operations, which is the same complexity than standard CNNs.

3. Dataset:

MovieLens: This data set contains 10000054 ratings and 95580 tags applied to 10681 movies by 71567 users of the online movie recommender service MovieLens.

Users were selected at random for inclusion. All users selected had rated at least 20 movies. Unlike previous MovieLens data sets, no demographic information is included.

[1] $T_j(\lambda) = \cos(j \cos^{-1}(\lambda))$ is an oscillating function on $[-1, 1]$ with j roots, $j + 1$ equally spaced extrema, and a frequency linearly dependent on j. Chebyshev polynomials form an orthogonal basis for the space of smooth functions on $[-1, 1]$ and are thus convenient to compactly represent spectral filters.

Each user is represented by an id, and no other information is provided.

Permalink to movie dataset.

Some pre-processing was required for creating sparse graphs of users (Wusers: [943x943]) and movies (Wmovies: [1682x1682]). Note that we took a split (943x1682) of the big matrix on which we are conducting our whole project (OTrainMat/OTestMat: [943x1682]).

4. Baseline:

Baseline of this task has been formulated, a matrix completion problem as a learnable diffusion process applied to the score values. Main approach of this task can be summarized in two steps:

- Extracting spatial feature from matrix variable. (Currently, used only full matrix approach not the factorized ones)
- Using LSTM RNN network for temporal part.

These architecture is summarized in Figure 2 and described in detail in the following sections.

4.1. Multi-Graph CNNs

To define convolution we first define graph Fourier transform row and column graphs of matrix. As we saw in literature survey section Fourier transform on graph is eigen-decomposition of Laplacian matrix of a graph into eigenvalues and eigenvectors. Analogously to classical Fourier Transform, the eigenvalues represent frequencies and eigenvectors form what is known as a graph Fourier basis.
In our setting, the analogy of the two-dimensional Fourier transform has the form,

$$\hat{\mathbf{X}} = \mathbf{\Phi}_r^\top \mathbf{X} \mathbf{\Phi}_c \tag{9}$$

where $\mathbf{\Phi}_c, \mathbf{\Phi}_r$ and $\mathbf{\Lambda}_c, \mathbf{\Lambda}_r$ denote the $n \times n$ and $m \times m$ eigenvector- and eigenvalue matrices of the column- and row-graph Laplacians $\mathbf{\Delta}_c, \mathbf{\Delta}_r$, respectively. The multi-graph version of the spectral convolution (6) is given by

$$\mathbf{X} \star \mathbf{Y} = \mathbf{\Phi}_r (\hat{\mathbf{X}} \circ \hat{\mathbf{Y}}) \mathbf{\Phi}_c^\top. \tag{10}$$

As we saw in literature survey section representing the filters as their spectral multipliers $\hat{\mathbf{Y}}$ would yield $\mathcal{O}(mn)$ parameters, which is infeasible in any practical application due to computational complexity. To get rid of this limitation we use the representation of the filters in Chebychev polynomial bases of degree p,

$$\tau_{\mathbf{\Theta}}(\tilde{\lambda}_c, \tilde{\lambda}_r) = \sum_{j,j'=0}^{p} \theta_{jj'} T_j(\tilde{\lambda}_c) T_{j'}(\tilde{\lambda}_r), \tag{11}$$

Algorithm 1 Full matrix completion model using RGCNN

input $m \times n$ matrix $\mathbf{X}^{(0)}$ containing initial values
1: **for** $t = 0 : T$ **do**
2: Apply the Multi-Graph CNN (12) on $\mathbf{X}^{(t)}$ producing an $m \times n \times q$ output $\tilde{\mathbf{X}}^{(t)}$ containing a q-dimensional feature vector for each matrix element.
3: **for** all elements (i, j) **do**
4: Apply RNN to feature vector $\tilde{\mathbf{x}}_{ij}^{(t)} = (\tilde{x}_{ij1}^{(t)}, \ldots, \tilde{x}_{ijq}^{(t)})$ producing the predicted incremental value $dx_{ij}^{(t)}$
5: **end for**
6: Update $\mathbf{X}^{(t+1)} = \mathbf{X}^{(t)} + \mathbf{dX}^{(t)}$
7: **end for**

where $\mathbf{\Theta} = (\theta_{jj'})$ is the $(p+1) \times (p+1)$ matrix of coefficients, i.e., $\mathcal{O}(1)$ parameters. The application of such filters to the matrix $\mathbf{X}$

$$\tilde{\mathbf{X}} = \sum_{j,j'=0}^{p} \theta_{jj'} T_j(\tilde{\mathbf{\Delta}}_r) \mathbf{X} T_{j'}(\tilde{\mathbf{\Delta}}_c) \tag{12}$$

Using this representation of filters (12) when convolution layer is applied to the $m \times n$ matrix $\mathbf{X}$ (single input channel), producing q outputs (i.e., a tensor of size $m \times n \times q$).

4.2. Matrix diffusion with RNN

In next step feature extracted from previous MGCNN are fed to LSTM-RNN implementing the score diffusion process. The input of the LSTM gate is given by the static features ex-tracted from the MGCNN, which can be seen as a dimensionality reduction of the original matrix in the space of the most meaningful and representative information.
Algorithm is described precisely in 1. The complexity of Algorithm 1 scales quadratically as $\mathcal{O}(m+n)$ due to the use of MGCNN.

4.3. Training

Training of the networks is performed by minimizing the loss

$$\ell(\mathbf{\Theta}, \boldsymbol{\sigma}) = \|\mathbf{X}_{\mathbf{\Theta},\boldsymbol{\sigma}}^{(T)}\|_{\mathcal{G}_r}^2 + \|\mathbf{X}_{\mathbf{\Theta},\boldsymbol{\sigma}}^{(T)}\|_{\mathcal{G}_c}^2 + \frac{\mu}{2} \|\mathbf{\Omega} \circ (\mathbf{X}_{\mathbf{\Theta},\boldsymbol{\sigma}}^{(T)} - \mathbf{Y})\|_{\mathrm{F}}^2. \tag{13}$$

Here, T denotes the number of diffusion iterations (applications of the RNN), and we use the notation $\mathbf{X}_{\mathbf{\Theta},\boldsymbol{\sigma}}^{(T)}$ to emphasize that the matrix depends on the parameters of the MGCNN (Chebyshev polynomial coefficients $\mathbf{\Theta}$) and those of the LSTM (denoted by $\boldsymbol{\sigma}$).

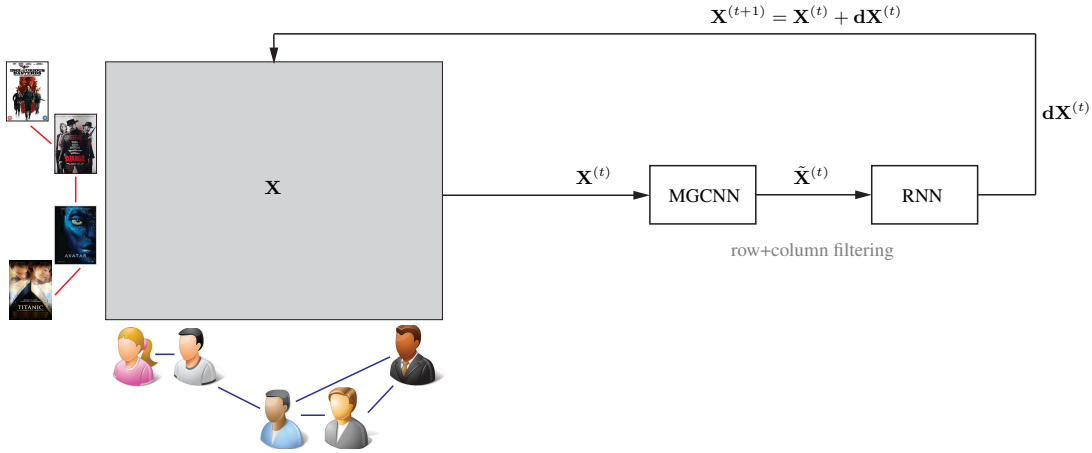

Figure 2. Recurrent GCNN (RGCNN) architecture using the full matrix completion model and operating simultaneously on the rows and columns of the matrix $\mathbf{X}$. The output of the Multi-Graph CNN (MGCNN) module is a q-dimensional feature vector for each element of the input matrix. The number of parameters to learn is $\mathcal{O}(1)$ and the learning complexity is $\mathcal{O}(mn)$.

Algorithm 2 Factorized matrix completion model using sRGCNN

input $m \times r$ factor $\mathbf{H}^{(0)}$ and $n \times r$ factor $\mathbf{W}^{(0)}$ representing the matrix $\mathbf{X}^{(0)}$

for $t = 0 : T$ **do**
  Apply the Graph CNN on $\mathbf{H}^{(t)}$ producing an $n \times q$ output $\tilde{\mathbf{H}}^{(t)}$.
  for $j = 1 : n$ **do**
    Apply RNN to feature vector $\tilde{\mathbf{h}}_j^{(t)} = (\tilde{h}_{j1}^{(t)}, \ldots, \tilde{h}_{jq}^{(t)})$ producing the predicted incremental value $dh_j^{(t)}$
  end for
  Update $\mathbf{H}^{(t+1)} = \mathbf{H}^{(t)} + \mathbf{dH}^{(t)}$
  Apply the Graph CNN on $\mathbf{W}^{(t)}$ producing an $m \times q$ output $\tilde{\mathbf{W}}^{(t)}$.
  for $i = 1 : m$ **do**
    Apply RNN to feature vector $\tilde{\mathbf{w}}_i^{(t)} = (\tilde{w}_{i1}^{(t)}, \ldots, \tilde{w}_{iq}^{(t)})$ producing the predicted incremental value $dw_i^{(t)}$
  end for
  Update $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \mathbf{dW}^{(t)}$
end for

In the factorized setting, we use the loss

$$\begin{aligned} \ell(\boldsymbol{\theta}_r, \boldsymbol{\theta}_c, \boldsymbol{\sigma}) &= \|\mathbf{W}^{(T)}_{\boldsymbol{\theta}_r,\boldsymbol{\sigma}}\|^2_{\mathcal{G}_r} + \|\mathbf{H}^{(T)}_{\boldsymbol{\theta}_c,\boldsymbol{\sigma}}\|^2_{\mathcal{G}_c} \\ &+ \frac{\mu}{2}\|\boldsymbol{\Omega} \circ (\mathbf{W}^{(T)}_{\boldsymbol{\theta}_r,\boldsymbol{\sigma}}(\mathbf{H}^{(T)}_{\boldsymbol{\theta}_c,\boldsymbol{\sigma}})^\top - \mathbf{Y})\|^2_{\mathrm{F}}. \end{aligned} \tag{14}$$

where $\boldsymbol{\theta}_c, \boldsymbol{\theta}_r$ are the parameters of the two GCNNs.

5. Main approach:

Summarizing main approach can be formulated into following steps:

- Input matrix X is constructed using the user and item (movie) graphs as unweighted 10-nearest neighbor-graphs in the space of user and movie features, respectively.
- Filters are learned using MGCNN over matrix X described in previous section. Chebyshev polynomials of order $p = 5$, outputting $k = 32$-dimensional features.
- Learned filters are fed into LSTM network with $T = 10$ diffusion steps.
- All the models were implemented in TensorFlow and trained using the Adam stochastic optimization algorithm with learning rate 10^{-3}.
- Output will be filled matrix representing the completed matrix

  The precise algorithm as presented in the previous page titled "Algorithm Full matrix completion model using RGCNN"

6. Evaluation Metric

We are targeting to increase accuracy by minimizing the RMSE i.e., Root Mean Squared Error. Hence RMSE is being used as the evaluation metric for our project. In official leaderboards for recommendation systems, RMSE, Mean Absolute Error (MAE) Normalized Discounted Cumulative Gain (NDCG) are some of the popular metrics in the

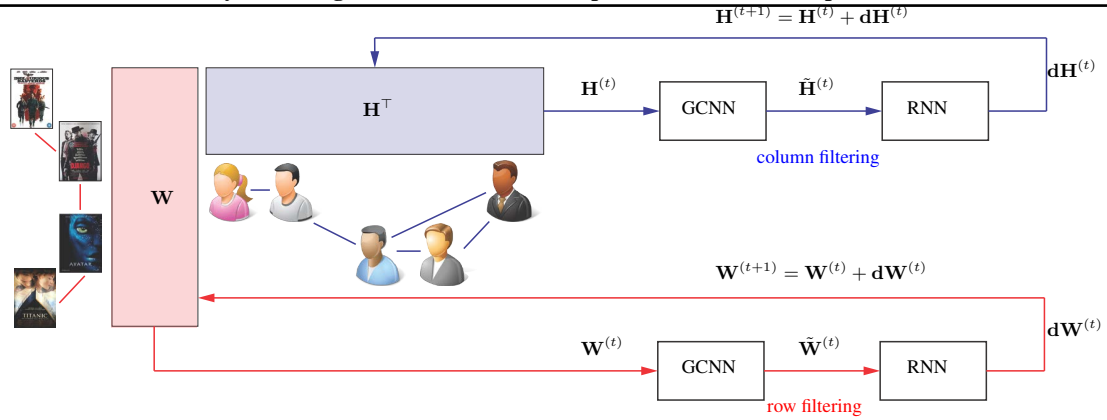

Figure 3. Separable Recurrent GCNN (sRGCNN) architecture using the factorized matrix completion model and operating separately on the rows and columns of the factors $\mathbf{W}$, $\mathbf{H}^\top$. The output of the GCNN module is a q-dimensional feature vector for each input row/column, respectively. The number of parameters to learn is $\mathcal{O}(1)$ and the learning complexity is $\mathcal{O}(m+n)$.

academic world. We are using RMSE (most used) since it is relatively easy to compute and offers easy interpretation and correlation with the performance of the recommendation system, along with easy comparison to other models to understand relative quality and capability of our model.

$$E_{\text{rmse}} = \sqrt{\sum_i (p_i - r_i)^2 / N} \tag{15}$$

where p_i is the predicted rating, r_i is the true rating. N is the number of validation samples.

7. Results & Analysis

We follow the setup of [28] using the two datasets i.e. the MovieLens and Netflix Synthetic dataset. Note that in factorized models, rank $r = 15$ and 10 were used for the synthetic and real datasets, respectively. In our the experiments, we used Chebyshev polynomials of order $p = 5$, outputting $k = 32$-dimensional features, $T = 10$ diffusion steps and LSTM cells with 32 features. We trained using the ADAM stochastic optimization algorithm with learning rate 10^{-3}.

7.1. Synthetic data

As per the observation, the RGCNN model achieves the best accuracy, followed by the separable RGCNN.

We tried the same experiment with only the column (users) graph provided. Here, the RGCNN can't be applied, while the sRGCNN has only one GCNN applied on the factor $\mathbf{H}$, and the other factor $\mathbf{W}$ is free. Table 2 summarizes the results of this experiment.

Table 1. Comparison of different matrix completion methods using *users+items graphs* in terms of number of parameters (optimization variables) and computational complexity order (operations per iteration). Rightmost column shows the RMS error on Synthetic dataset.

METHOD	PARAMETERS	COMPLEXITY	RMSE
GMC	mn	mn	0.3693
GRALS	$m+n$	$m+n$	0.0114
RGCNN	$\mathbf{1}$	$\mathbf{mn}$	**0.0053**
sRGCNN	$\mathbf{1}$	$\mathbf{m+n}$	**0.0106**

Table 2. Comparison of different matrix completion methods using *users graph only* in terms of number of parameters (optimization variables) and computational complexity order (operations per iteration). Rightmost column shows the RMS error on Synthetic dataset.

METHOD	PARAMETERS	COMPLEXITY	RMSE
GRALS	$m+n$	$m+n$	0.0452
sRGCNN	$\mathbf{m}$	$\mathbf{m+n}$	**0.0362**

7.2. Real data

We evaluated the approach on the MovieLens dataset. For the MovieLens dataset we have the user and item (movie) graphs as unweighted 10-nearest neighbor graphs in the space of user and movie features, respectively.

Tables 3 summarizes the performance of different methods. RGCNN outperforms the competitors in all the experiments.

Table 3. Performance (RMS error) of different matrix completion methods on the MovieLens dataset.

METHOD	RMSE
GLOBAL MEAN	1.154
USER MEAN	1.063
MOVIE MEAN	1.033
MC[2]	0.973
IMC[3]	1.653
GMC[7]	0.996
GRALS[8]	0.945
sRGCNN	**0.929**

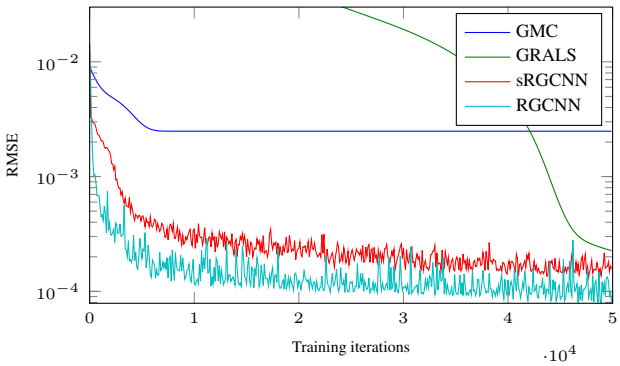

Figure 4. Convergence rates of the tested algorithms over the Synthetic Netflix dataset.

8. Error Analysis

The first architecture works on the full matrix model producing better accuracy but requiring higher complexity. The second uses factorized matrix model, offering better scalability at the expense of slight reduction of accuracy.

The RGCNN model achieves the best accuracy, followed by the separable RGCNN. Different diffusion time steps of these two models are visualized in Figure 5. Figure 4 shows the convergence rates of different methods.

9. Conclusion

In this paper, we presented a new deep learning approach for matrix completion based on a specially designed multi-graph convolutional neural network architecture. Among the key advantages of our approach compared to traditional methods is its low computational complexity and constant number of degrees of freedom independent of the matrix size. We showed that the use of deep learning for matrix completion allows to achieve decent results in movie recommendation.

10. Future Work

The model could be improved by using transformers in the loop instead of the RNN. Another major add-on could be discrete Synaptic Kernel Filters to further optimize the individual scores.

The major challenge in building this system has been deeplearing on non-euclidean space (fundamental aspect of Geometric Deep learning) as it inherently makes the model more complex. We are trying to exploit the local stationary structures on user/item graphs and hence meaningful statistical graph-structured patterns.

The graph representation definitely makes it easy to visualize, however the implementation can be further simplified for easy implementation and model upgradations.

11. Code

Implementation (python code) link to github is https://github.com/shashankchaurasia-code/stanford_cs221_project_spring2021

References

[1] Geometric Matrix Completion with Recurrent Multi-Graph Neural Networks by Federico Monti, Michael M. Bronstein, Xavier Bresson

[2] Candes, E. and Recht, B. Exact matrix completion via convex optimization. Comm. ACM, 55(6):111–119, 2012.

[3] Jain, P. and Dhillon, I. S. Provable inductive matrix completion. arXiv:1306.0626, 2013.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv:1409.1556, 2014.

[5] "Geometric deep learning on graphs and manifolds using mixture model CNNs," by Bronstein, M. et al.pndopt,

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[7] Kalofolias, V., Bresson, X., Bronstein, M. M., and Vandergheynst, P. Matrix completion on graphs. arXiv:1408.1717, 2014.

[8] Rao, N., Yu, H.-F., Ravikumar, P. K., and Dhillon, I. S. Collaborative filtering with graph information: Consistency and scalable methods. In Proc. NIPS, 2015.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in Proc. NIPS, 2014.

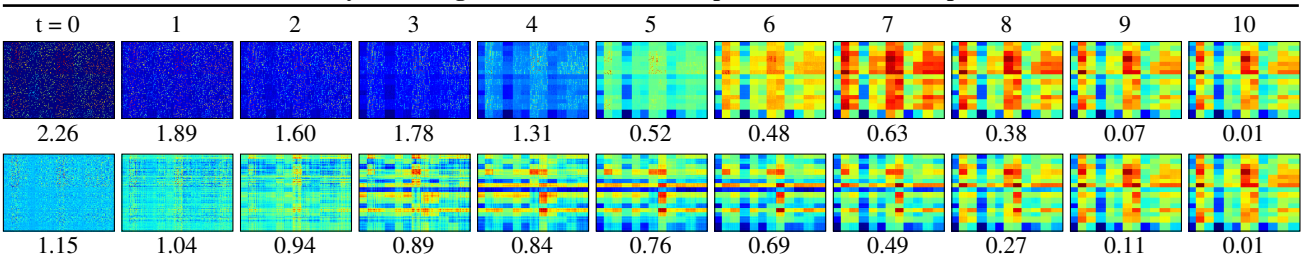

Figure 5. Evolution of the matrix $\mathbf{X}^{(t)}$ with our architecture using full matrix completion model RGCNN (top) and factorized matrix completion model sRGCNN (bottom). Numbers indicate the RMS error.

[10] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in Proc. ISCAS, 2010.

[11] D. Cires¸an, U. Meier, J. Masci, and J. Schmidhuber, "A committee of neural networks for traffic sign classification," in Proc. IJCNN, 2011.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in Proc. NIPS, 2012.

[13] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," Trans. PAMI, vol. 35, no. 8, pp. 1915–1929, 2013.

[14] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in Proc. CVPR, 2014.

[15] Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.

[16] Jain, P. and Dhillon, I. S. Provable inductive matrix completion. arXiv:1306.0626, 2013.

[17] Jamali, M. and Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proc. Recommender Systems, 2010.

[18] Kalofolias, V., Bresson, X., Bronstein, M. M., and Vandergheynst, P. Matrix completion on graphs. arXiv:1408.1717, 2014.

[19] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014.

[20] Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907, 2016.

[21] Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. Computer, 42(8): 30–37, 2009.

[22] Kuang, D., Shi, Z., Osher, S., and Bertozzi, A. A harmonic extension approach for collaborative ranking. arXiv:1602.05127, 2016.

[23] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradientbased learning applied to document recognition. Proc. IEEE, 86(11):2278–2324, 1998.

[24] Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. arXiv:1511.05493, 2015.

[25] Ma, H., Zhou, D., Liu, C., Lyu, M., and King, I. Recommender systems with social regularization. In Proc. Web Search and Data Mining, 2011.

[26] Masci, J., Boscaini, D., Bronstein, M. M., and Vandergheynst, P. Geodesic convolutional neural networks on Riemannian manifolds. In Proc. 3DRR, 2015.

[27] Miller, B. N. et al. MovieLens unplugged: experiences with an occasionally connected recommender system. In Proc. Intelligent User Interfaces, 2003.

[28] N. Rao, H.-F. Yu, P. K. Ravikumar, and I. S. Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In Proc. NIPS, 2015.

[29] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. IEEE Trans. Neural Networks, 20(1):61–80, 2009.

[30] Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. Structured sequence modeling with graph convolutional recurrent networks. arXiv:1612.07659, 2016.