A Survey Over UltraGCN and its Variants for Recommendation

Jerry Yan [1]

Abstract

Graph convolutional networks (GCNs) have significantly advanced collaborative-filtering by leveraging higher-order connectivity in user–item graphs. Among these, LightGCN removes feature transformations and nonlinearities to yield an efficient, layer-wise propagation (or message passing), while UltraGCN further approximates the infinite-layer limit of LightGCN via a closed-form optimization scheme. In this paper, we first review the architectures of LightGCN and UltraGCN, meanwhile providing a rigorous proof that UltraGCN's infinite-layer propagation converges to the fixed-point solution used in practice. To assess the impact of the graph-constraint term, we conduct an empirical study on standard benchmarks: we compare (i) truncated SVD, (ii) UltraGCN without any graph constraint loss, (iii) UltraGCN with a modified graph constraint loss omitting the log-sigmoid activation, and (iv) the original UltraGCN. Our experiments demonstrate that (a) UltraGCN's closed-form propagation significantly outperforms truncated SVD baseline, and (b) both the infinite-layer graph constraint and its log-sigmoid activation are essential for achieving peak recommendation accuracy. These results elucidate both the theoretical foundations and practical benefits of infinite-layer message passing in GCN-based recommender systems.

1. Introduction

Recommender systems lie at the heart of many online platforms, guiding users through vast catalogs of items by predicting their preferences. Traditional collaborative filtering methods such as matrix factorization and truncated singular value decomposition (SVD) exploit low-rank structure in the user–item interaction matrix, but they struggle to incorporate higher-order "friend-of-friend" relations that can be crucial for capturing subtle patterns of taste.

Graph convolutional networks (GCNs) have emerged as a powerful paradigm for recommendation by viewing the user–item interactions as a bipartite graph and propagating embeddings through its connectivity structure. LightGCN (He et al., 2020) demonstrated that one can remove feature transformations and nonlinearities from standard GCNs, yielding a simple yet highly effective layer-wise propagation scheme that aggregates multi-hop signals efficiently. However, stacking many layers in LightGCN both increases computation and risks over-smoothing, where all node embeddings collapse to a single point.

UltraGCN (Mao et al., 2021) addresses this by deriving a closed-form approximation to the infinite-depth limit of LightGCN's propagation. By solving a fixed-point equation directly, UltraGCN omits deep stacking entirely, offering constant-time message passing and impressive empirical performance. Despite its practical success, a concise, rigorous proof on the convergence of the infinite-layer limit has not appeared in the literature. Moreover, the role of the graph-constraint loss—particularly its log-sigmoid activation that down-weights popular nodes—has only been evaluated in the original UltraGCN setup, leaving open questions about its necessity and potential variants.

In this paper, we fill these gaps by offering both theoretical and empirical contributions:

- We review the LightGCN and UltraGCN architectures, and present a self-contained proof that UltraGCN's infinite-layer propagation converges, leading to its closed-form fixed-point solution.
- We establish a controlled empirical study on standard benchmarks, comparing:
  1. Truncated SVD as a classical low-rank baseline.
  2. UltraGCN without any graph-constraint loss.
  3. UltraGCN with a modified graph-constraint loss that omits the log-sigmoid activation.
  4. The original UltraGCN with its log-sigmoid-activated graph-constraint term.

The rest of the paper is organized as follows. In Section 2 we introduce notation and revisit LightGCN. In Section 3 we prove the convergence of UltraGCN's infinite-layer limit and give an overview on its architecture. In Section 4 we describe our experimental setup and results, and finally conclude in Section 5.

2. Background

In this section we introduce our notation and review the LightGCN framework on which UltraGCN is built.

2.1. Notation and Setup

Let $\mathcal{U} = \{u_1, \ldots, u_{|\mathcal{U}|}\}$ be the set of users and $\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{V}|}\}$ the set of items. We view interactions as a bipartite graph $\mathcal{G} = (\mathcal{U} \cup \mathcal{V}, \mathcal{E})$, where

$$\mathcal{E} = \big\{(u, i) : \text{user } u \text{ has interacted with item } i\big\}.$$

Let $A \in \{0,1\}^{(|\mathcal{U}|+|\mathcal{V}|)\times(|\mathcal{U}|+|\mathcal{V}|)}$ be the adjacency matrix of $\mathcal{G}$. Further let $\tilde{A} = A + I$ be the adjacency matrix with self-loop added, and let

$$\tilde{D} = \mathrm{diag}\big(\tilde{A}\mathbf{1}\big)$$

be its degree matrix. Let d_u, d_i be the degree of user u and item i in $\mathcal{G}$. We further denote by

$$\mathbf{e}_u^{(0)} \in \mathbb{R}^n, \quad \mathbf{e}_i^{(0)} \in \mathbb{R}^n$$

the initial embeddings of user u and item i, respectively, assembled into

$$E^{(0)} = [\,\mathbf{e}_1^{(0)}, \ldots, \mathbf{e}_{|\mathcal{U}|+|\mathcal{V}|}^{(0)}]^\top.$$

We further assume $\mathcal{G}$ to be a strongly connected graph. In practice when this is not the case, it is reasonable to drop the isolated nodes (users or items with zero interactions), and only focus on the larger component.

2.2. Revisiting LightGCN

LightGCN (He et al., 2020) simplifies traditional GCNs by removing feature transformations and nonlinearities, retaining only neighborhood aggregation. At layer $\ell + 1$, embeddings are updated via

$$E^{(\ell+1)} \;=\; \underbrace{\tilde{D}^{-\frac{1}{2}}\,\tilde{A}\,\tilde{D}^{-\frac{1}{2}}}_{:=\hat{A}}\; E^{(\ell)}\,. \tag{1}$$

Equivalently, for a user node u,

$$\begin{aligned}(\hat{A}E^{(\ell+1)})_{u,*} &= \sum_k \hat{A}_{u,k} E^{(\ell)}_{k,*} \\ &= \sum_k \hat{A}_{u,k}\mathbf{e}_k^{(\ell)} \\ &= \hat{A}_{u,u}\mathbf{e}_u^{(\ell)} + \sum_{k\neq u} \hat{A}_{u,k}\mathbf{e}_k^{(\ell)}\end{aligned}$$

Observe that $\tilde{A}_{u,k} \in \{0,1\}$, given $\tilde{A}_{u,k} = 1$:

$$\begin{aligned}\hat{A}_{u,k} &= \sum_{i,j} (\tilde{D}^{-\frac{1}{2}})_{u,i}\,\tilde{A}_{i,j}\,(\tilde{D}^{-\frac{1}{2}})_{j,k} \\ &= (\tilde{D}^{-\frac{1}{2}})_{u,u}\,\tilde{A}_{u,k}\,(\tilde{D}^{-\frac{1}{2}})_{k,k} \\ &= \frac{1}{\sqrt{d_u+1}\sqrt{d_i+1}}\end{aligned}$$

So the user embedding is updated as

$$\mathbf{e}_u^{(\ell+1)} = \frac{1}{d_u+1}\mathbf{e}_u^{(\ell)} + \sum_{i\in\mathcal{N}(u)} \frac{1}{\sqrt{d_u+1}\sqrt{d_i+1}}\,\mathbf{e}_i^{(\ell)}\,, \tag{2}$$

with the same for items. These embeddings are then trained under a pairwise ranking loss (e.g. BPR) to separate observed from unobserved interactions.

This simple propagation captures high-order connectivity (up to K hops) in the user–item graph without costly nonlinear transformations. However, deep stacking ($K \gg 1$) may incur excessive computation and lead to over-smoothing, motivating UltraGCN's infinite-layer approximation design.

3. UltraGCN: Infinite-Layer Approximation

UltraGCN derives a closed-form approximation to LightGCN's deep propagation by taking the limit as the number of layers $K \to \infty$.

3.1. Convergence of the Infinite-Depth Limit

LightGCN updates embeddings at layer $\ell + 1$ via

$$E^{(\ell+1)} \;=\; \hat{A}\,E^{(\ell)}.$$

Hence after K hops,

$$E^{(K)} \;=\; \hat{A}^K\,E^{(0)},$$

Therefore, to approximate infinite-layer message passing, it must first be shown such embedding $E^{(\infty)}$ converges.

Theorem 3.1. *The infinite-layer embedding*

$$E^{(\infty)} = \lim_{k\to\infty} \hat{A}^k\,E^{(0)}$$

exists.

Proof. First, observe that $\hat{A} = \tilde{D}^{-\frac{1}{2}}\,\tilde{A}\,\tilde{D}^{-\frac{1}{2}}$ is a real, symmetric matrix, so by eigen-decomposition there is

$$\hat{A} = U\,\Lambda\,U^\top$$

where

$$U^\top U = I, \quad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

Next, observe that $\hat{A}$ is primitive, that is, it is

1. non-negative,
2. aperiodic (enforced by the self-loop),
3. and irreducible (by the strongly connected assumption of user-item graph $\mathcal{G}$).

Remark 3.2. All statements of the Perron–Frobenius theorem for positive matrices remain true for primitive matrices.

Let $v = \tilde{D}^{1/2}\mathbf{1}$, we claim it is an eigenvector of $\hat{A}$ with eigenvalue 1:

$$\begin{aligned}\hat{A}v &= \tilde{D}^{-1/2}(A+I)\tilde{D}^{-1/2}\tilde{D}^{1/2}\mathbf{1}\\ &= \tilde{D}^{-1/2}(A+I)\mathbf{1}\\ &= \tilde{D}^{-1/2}\tilde{D}\mathbf{1}\\ &= D'^{1/2}\mathbf{1}\\ &= v\end{aligned}$$

So using Remark 3.2 and Perron-Forbenius, as v is positive (thanks to the self-loops), it is the Perron vector corresponding to $\lambda_1 = 1$. So

$$\mid \lambda_i \mid < 1, \quad \forall i \neq 1.$$

As such:

$$\hat{A}^k = U\,\Lambda^k\,U^\top = \sum_{i=1}^{n} \lambda_i^k u_i u_i^\top,$$

and as $k \to \infty$, largest eigenvalue $\lambda_1 = 1$ dominates, such that

$$\lim_{k\to\infty} \hat{A}^k = u_1 u_1^\top,$$

where u_1 is the normalized eigenvector $\frac{v}{\|v\|_2}$. So

$$\begin{aligned}(\lim_{k\to\infty} \hat{A}^k)_{ij} &= \frac{u_1^{(i)} u_1^{(j)}}{\|u_1\|_2^2}\\ &= \frac{\sqrt{d_i+1}\sqrt{d_j+1}}{\sum_k d_k + 1}\\ &= \frac{\sqrt{d_i+1}\sqrt{d_j+1}}{2m+n}\end{aligned}$$

where m, n are the number of edges and nodes, repsectively. □

3.2. Closed-Form Message-Passing Scheme

As $E^{(\infty)}$ exists, from the fixed-point condition

$$E^{(\infty)} = \hat{A}E^{(\infty)}$$

and Equation (2), we similarly have

$$\mathbf{e}_u^{(\infty)} = \frac{1}{d_u+1}\mathbf{e}_u^{(\infty)} + \sum_{i\in\mathcal{N}(u)} \frac{1}{\sqrt{d_u+1}\sqrt{d_i+1}}\,\mathbf{e}_i^{(\infty)},$$

by simple algebra we derive the following closed-form update rule for ultraGCN:

$$\mathbf{e}_u = \sum_{i\in\mathcal{N}(u)} \beta_{u,i}\mathbf{e}_i, \quad \beta_{u,i} = \frac{1}{d_u}\sqrt{\frac{d_u+1}{d_i+1}} \tag{3}$$

3.3. Graph-Constraint Loss

To encourage the final embedding be similar to the fixed-point embeddings, we maximize their cosine similarity:

$$\max(\sum_{i\in\mathcal{N}(u)} \beta_{u,i}\mathbf{e}_i)^\top \mathbf{e}_u, \quad \forall u \in \mathcal{U}$$

which becomes

$$\max \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{N}(u)} \beta_{u,i}\mathbf{e}_i^\top \mathbf{e}_u. \tag{4}$$

To prevent focusing on only a small set of popular user/item embeddings during the maximization process, and for ease of optimization with bounded gradients, we apply the log-sigmoid activation:

$$\max \sum_{u\in\mathcal{U}} \sum_{i\in\mathcal{N}(u)} \beta_{u,i} \log\sigma(\mathbf{e}_i^\top \mathbf{e}_u). \tag{5}$$

As such, the graph constraint loss is formulated as:

$$\mathcal{L}_{\text{graph}} = -\sum_{(u,i)\in\mathcal{E}} \beta_{u,i}\,\log\sigma\big(\mathbf{e}_u^\top \mathbf{e}_i\big). \tag{6}$$

However, this loss suffers from the over-smoothing problem, where user and item embeddings collapse to the same embedding. In traditional lightGCN, usually a small number of message-passing layers is fixed to circumvent this issue (e.g. 2-4 layers) (He et al., 2020). However, in the case of infinite-layer approximation, there is no such option.

To counteract over-smoothing, a negative-sampling term is added such that non-interactions and negative-interactions (u, j) are encouraged to have dissimilar embeddings. This leads to the new graph loss:

$$\mathcal{L}_{\text{graph}} = -\sum_{(u,i)\in\mathcal{E}^+} \beta_{u,i}\,\log\sigma\big(\mathbf{e}_u^\top \mathbf{e}_i\big) - \sum_{(u,j)\in\mathcal{E}^-} \beta_{u,j}\,\log\sigma\big(-\mathbf{e}_u^\top \mathbf{e}_j\big) \tag{7}$$

where $\mathcal{E}^+$ are the positive interactions and $\mathcal{E}^-$ are the negative and non-interactions.

3.4. Optimization Objective

To enforce the quality of recommendation, we can further apply either the Bayesian Personalized Ranking (BPR) loss or the Binary Cross Entropy (BCE) loss. For this paper, we use the BCE loss

$$\mathcal{L}_{\text{BCE}} = -\sum_{(u,i)\in\mathcal{E}^+} \log\sigma\left(\mathbf{e}_u^\top \mathbf{e}_i\right) - \sum_{(u,j)\in\mathcal{E}^-} \log\sigma\left(-\mathbf{e}_u^\top \mathbf{e}_j\right) \tag{8}$$

Combining the losses to have

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \lambda\,\mathcal{L}_{\text{graph}}, \tag{9}$$

with hyperparameter $\lambda > 0$ controlling the strength of the infinite-layer approximation constraint.

4. Experimental Setup and Results

In this section we describe our datasets, baselines, metrics, and implementation details, then present the quantitative results.

4.1. Datasets and Preprocessing

We evaluate on the MovieLens100K dataset. Following standard practice, we:

- Filter out users and items with fewer than 5 interactions.
- Extract the largest connected component of the user–item bipartite graph.
- Split interactions per user by randomly holding out one positive for testing; the rest are used for training.

4.2. Baselines and Variants

We compare four methods:

1. **Truncated SVD**: rank-20 approximation of the interaction matrix.
2. **UltraGCN w/o graph constraint**: infinite-layer propagation only, trained with BPR loss.
3. **UltraGCN (modified constraint)**: original graph-constraint term but omitting the $\log\sigma(\cdot)$ weighting.
4. **UltraGCN (original)**: full model with log-sigmoid-weighted graph constraint.

4.3. Evaluation Metrics

As metrics for performance, we report:

- **Hit Ratio@10 (HR@10)**: fraction of held-out items appearing in the top–10 recommendations.
- **NDCG@10**: normalized discounted cumulative gain at rank 10, computed as $\text{NDCG}_{10} = \sum_{i=1}^{10} \frac{2^{\text{rel}_i}-1}{\log_2(i+1)}$, where $\text{rel}_i \in \{0,1\}$ is whether the item appears at the ith rank.

4.4. UltraGCN General Recommendation Quality

To assess the general recommendation quality of original UltraGCN, we first compare it with a standard SVD baseline.

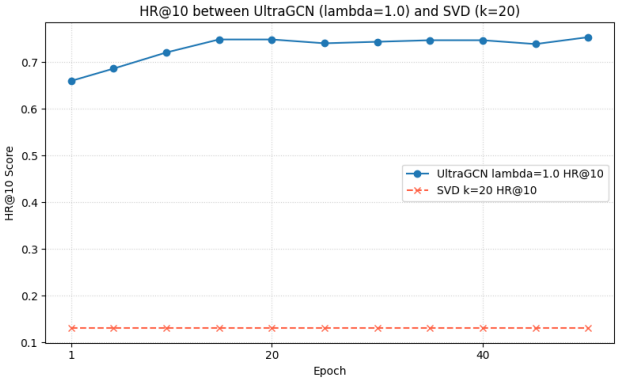

Figure 1. UltraGCN vs. SVD HR@10 Comparison

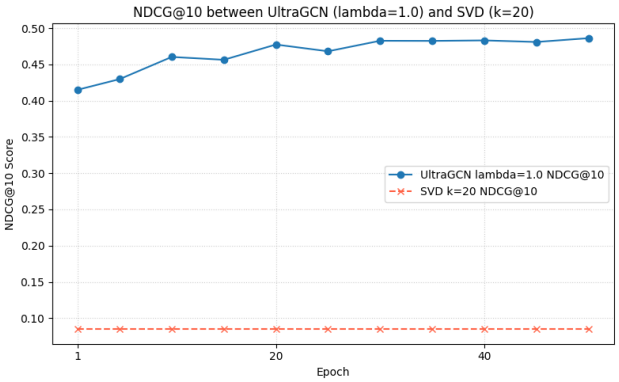

Figure 2. UltraGCN vs. SVD NDCG@10 Comparison

4.5. Graph Constraint Loss and Log-Sigmoid Activation Performance Gain

We next evaluate the impact of $\mathcal{L}_{\text{graph}}$. In this section, we compare different variants of UltraGCN having

1. no graph constraint loss ($\lambda = 0.0$), and

2. an explicit graph loss without the log-sigmoid activation. Recall from Equation (4) the original loss function, which explicitly enforces the infinite-layer constraint. Here, we modify the loss function to

$$\mathcal{L}'_{\text{graph}} = -\sum_{(u,i)\in\mathcal{E}^+} \beta_{u,i}\, \mathbf{e}_u^\top \mathbf{e}_i - \sum_{(u,j)\in\mathcal{E}^-} \beta_{u,j}\, (-\mathbf{e}_u^\top \mathbf{e}_j)$$

to incorporate negative sampling, and

3. the original UltraGCN ($\lambda = 1.0$).

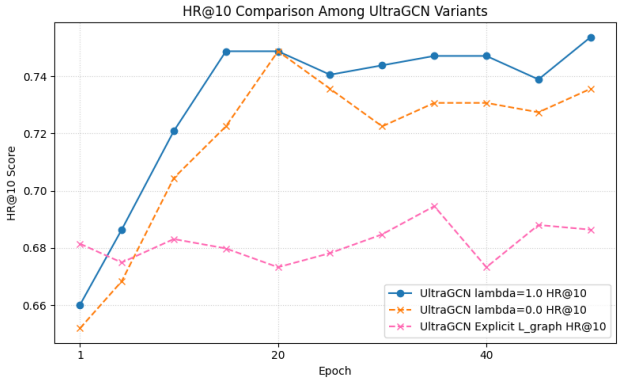

Figure 3. UltraGCN Variants on HR@10

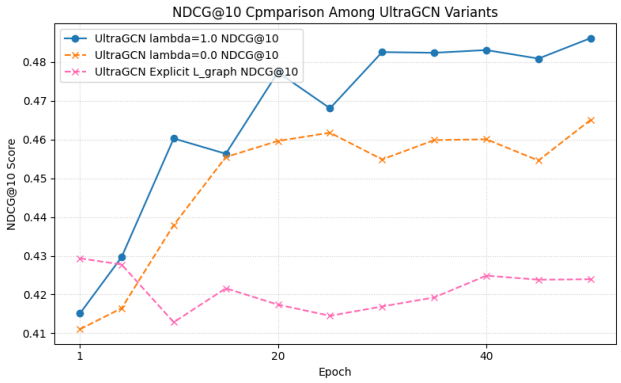

Figure 4. UltraGCN Variants on NDCG@10

4.6. Results

Table 1 reports HR@10 and NDCG@10 on the test set.

Table 1. Recommendation performance on MovieLens100K.

Method	HR@10	NDCG@10
Truncated SVD	0.1304	0.0847
UltraGCN w/o graph constraint	0.7356	0.4651
UltraGCN (modified constraint)	0.6864	0.4239
UltraGCN (original)	**0.7537**	**0.4862**

5. Conclusion

In this paper, we have revisited the LightGCN framework and provided a concise, self-contained proof that its infinite-layer limit converges to the closed-form fixed-point solution used by UltraGCN. We then carried out a systematic empirical study on MovieLens100K, comparing (i) truncated SVD, (ii) UltraGCN without any graph-constraint loss, (iii) UltraGCN with a modified (no log-sigmoid) constraint, and (iv) the original UltraGCN with its log-sigmoid-weighted graph-constraint term. Our results demonstrate three key findings:

- **Infinite-layer propagation outperforms classical low-rank methods**. All UltraGCN variants beat truncated SVD (rank-20) by a comfortable margin in both HR@10 and NDCG@10, confirming the value of incorporating arbitrarily high-order connectivity.
- **Graph-constraint regularization boosts accuracy**. Adding log-sigmoid activated graph-constraint on top of the BCE loss yields further improvements over the unconstrained infinite-layer model.
- **Log-sigmoid weighting is essential**. The original UltraGCN's use of a $\log\sigma(\cdot)$ activation scheme provides a clear additional gain over the unactivated variant. The performance drop in the unactivated variant is most likely attributed by the difference in magnitude between $\mathcal{L}'_{\text{graph}}$ and $\mathcal{L}_{\text{BCE}}$, which introduces instability and causes $\mathcal{L}'_{\text{graph}}$ to dominate the optimization process.

Beyond accuracy, UltraGCN retains constant-time propagation complexity, making it an attractive choice for large-scale recommender systems where deep stacking is impractical. In future work, it would be interesting to extend the fixed-point analysis to dynamic or heterogeneous graphs, and evaluate on larger industry-scale datasets.

References

He, X. et al. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proc. of the 43rd Intl. ACM SIGIR*, pp. 639–648, 2020.

Mao, K., Zhu, J., et al. Ultragcn: Ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1253–1262, 2021.