The Role of User-Item Connectivity in CTR Prediction

Anonymous Author(s)

Abstract

Click-through rate (CTR) prediction is a central problem in recommendation systems. Modern CTR models are largely feature-centric. They encode users, items, and contextual variables as independent feature fields, even though the same training data also define a user-item interaction graph. Graph-based methods can exploit this structure, but less is known about the role of connectivity itself. These efforts do not ask why connectivity helps, when it produces reliable gains, and which user groups receive them. This paper studies user-item connectivity as the object of analysis rather than as another architectural component. We use a deliberately minimal message-passing (MP) operator as a controlled intervention that enriches user and item representations with graph-based information while leaving the CTR architecture unchanged. We first show that MP smooths learned representations over the interaction graph and reduces sensitivity to small input perturbations, with stronger effects for nodes with more observed interactions. Experiments across datasets with different interaction densities show that connectivity is not uniformly beneficial. Denser graphs produce more consistent gains, while sparse graphs benefit less consistently. We further find that connectivity-aware embeddings can be efficiently integrated by fine-tuning existing CTR models. Finally, we introduce the Marginal Cost of Improvement (MCI), which measures the effective effort associated with improving each user group. Across our experiments, MP shifts more predictive improvement toward under-served user groups, leading to a more balanced distribution of predictive performance.

CCS Concepts

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Supervised learning*.

Keywords

Click-through rate prediction, message passing, recommender systems, graph neural networks

ACM Reference Format:

Anonymous Author(s). 2026. The Role of User-Item Connectivity in CTR Prediction. In *Proceedings of the 35th International ACM Conference on Knowledge and Information Management (CIKM '26), November 7–11, 2026, Rome, Italy*. ACM, New York, NY, USA, 11 pages.

1 Introduction

Click-through rate (CTR) prediction is a central problem in recommendation systems, estimating the probability that a user interacts with an item [3, 13, 18, 24, 33]. Modern CTR models are largely feature-centric, in that they encode users, items, and contextual variables as independent feature fields, then apply increasingly expressive interaction modules to predict engagement [5, 16, 19, 26, 33]. This design has produced strong aggregate performance, but it also treats user and item identifiers primarily as independent features. The same training data, however, also defines a user-item interaction graph. Observed clicks, ratings, and other engagements connect users to items, and this connectivity carries collaborative information that is not explicit when users and items are represented only as independent features.

Graph-based recommendation methods exploit this structure by propagating information over the user-item graph [10, 34], and several CTR models have also introduced graph-based components [15, 20, 36]. Most of this work asks whether graph structures can improve prediction, usually by proposing a stronger architecture or reporting aggregate gains. However, *less is known about the role of connectivity itself.* In particular, three questions remain under-characterized: *(i)* Why should user-item connectivity stabilize or improve CTR representations? *(ii)* Under what graph-density conditions should such connectivity produce reliable gains? *(iii)* When gains occur, which user groups receive them?

This paper studies user-item connectivity as the object of analysis rather than as another architectural component. Our goal is not to introduce a new graph-based CTR model. Instead, we use a deliberately simple message-passing (MP) operator as a controlled intervention. The operator enriches the learned representations of users and items with graph-based information while leaving CTR architectures unchanged. This lets us isolate the effect of connectivity from the effect of additional model expressiveness. In this sense, *MP functions as a probe by allowing us to ask when the interaction graph is a useful resource for CTR prediction, why it helps, and how its benefits are distributed.*

We first provide a stability-based account of connectivity-aware representation learning. We show that MP smooths learned representations over the interaction graph and reduces sensitivity to small perturbations in input features. This effect can be characterized through a reduction in a Lipschitz constant, with stronger smoothing for nodes with more observed interactions (Theorem 4.1). The analysis suggests that connectivity should be most useful when the graph contains enough reliable interaction structure to support stable aggregation.

We then test this prediction empirically across datasets with different interaction densities. The results show that connectivity is not uniformly beneficial. On denser interaction graphs, MP yields more consistent improvements across CTR architectures. On sparse graphs, the effects are less consistent. We also evaluate two integration strategies and find that fine-tuning existing CTR models with connectivity-aware embeddings can often match or outperform training MP-enhanced models from scratch.

Finally, we examine how connectivity changes the distribution of predictive gains across users. Aggregate metrics such as AUC indicate absolute gains, but they do not reveal how much effective effort is directed toward different user groups. We therefore stratify users by activity level and define under-served users as those whose predictive performance is lower relative to other groups. To quantify how MP reallocates improvement, we introduce the Marginal Cost of Improvement (MCI), which measures the effective effort associated with improving each group. The intuition is that equal absolute AUC gains are not equally difficult; in other words,

improving a group already performing close to the performance ceiling requires more effort than a group far below the ceiling. MCI captures this difference by scaling performance changes according to each group's baseline performance and the remaining room for improvement. Across our experiments, MP often shifts more of the improvement toward the under-served user groups. These results show that connectivity-aware representations can redistribute predictive gains across activity-defined user segments, reducing group-level performance imbalance.

In summary, our contributions are:

- We formulate user-item connectivity as an object of analysis for CTR prediction. Using a minimal MP operator as a controlled intervention, we isolate the effect of graph connectivity from the feature-interaction architecture of downstream CTR models.
- We provide a theoretical stability-based explanation for why connectivity can help. Our analysis shows that MP smooths representations over the interaction graph and reduces sensitivity to input perturbations, with a stronger effect for higher-degree users and items.
- We empirically characterize when connectivity is useful. Across multiple CTR architectures and datasets with different interaction densities, we show that sufficient graph connectivity is a key condition for reliable gains, while sparse interaction graphs produce less consistent gains. We further show that connectivity-aware embeddings can be incorporated through fine-tuning, offering a practical integration path for existing CTR systems.
- We study how connectivity affects the distribution of gains across users by proposing the Marginal Cost of Improvement (MCI) metric. Using this metric, we show that MP can shift predictive improvement toward user segments that are initially less well served by the feature-centric CTR models, leading to a more balanced distribution of predictive performance.

2 Related Work and Positioning

CTR Prediction. Early CTR prediction methods focused on modeling low- and high-order feature interactions between user-item pairs and contextual features, including factorization-based models [23], deep neural models [5, 9, 16], and more recent feature-interaction architectures [1, 19, 26, 27, 32, 33, 38]. These methods have substantially improved predictive accuracy by designing stronger interaction functions over encoded features. However, they typically encode user and item IDs independently through lookup tables, leaving the observed user-item interaction graph outside the representation-learning process.

Message Passing for Recommendation. GNNs and MP-based models provide a natural way to exploit relational structure [4, 6, 14, 28]. In recommendation, graph-based collaborative filtering methods such as NGCF [34] and LightGCN [10] propagate information over user-item interaction graphs, while later works continue to improve graph-based recommendation architectures [17, 22, 30, 35, 37]. Some CTR studies also incorporate graph-based mechanisms [15, 20, 36], but they primarily aim to design stronger predictive frameworks, often through feature-level graph structures or model innovations.

Our Position. Our goal is different. We do not propose a new CTR framework or another MP architecture. Instead, we formulate user-item connectivity as an object of analysis for CTR prediction, and use a deliberately minimal MP operator as an analytical probe to isolate the effect of user-item connectivity on CTR prediction. This allows us to study how connectivity reshapes model behaviors when density enables reliable gains and how MP redistributes predictive improvement across user segments. Thus, the contribution of this work is not architectural novelty, but a systematic analysis with a minimal MP operator to study the role of connectivity in CTR prediction.

3 The Prevailing Paradigm of CTR Prediction and Message Passing

Click-through Rate Prediction. Click-through rate (CTR) prediction is defined as predicting the interaction likelihood between a user and an item given the user ID, item ID, and optional contextual features. Formally, we denote IDs of user i and item j as x_i and x_j respectively, and the contextual features of the interaction in between as $\mathbf{c}_{ij} \in \mathbb{R}^{d^c}$, where d^c refers to the contextual feature dimension. Let the encoder of user/item IDs be $f(\cdot) : \mathbb{R} \to \mathbb{R}^d$, where d refers to the latent dimension of encoded ID embeddings.

The traditional encoder $f(\cdot)$ in CTR models is a look-up table encoder, where $f(x_i)$ is the x_i-th entry in a matrix $E \in \mathbb{R}^{(|\mathcal{U}|+|\mathcal{I}|)\times d}$, where $\mathcal{U}$ is the user set, $\mathcal{I}$ is the item set, and d is the embedding dimension. This table encoder, illustrated in Figure 1 (a), does not utilize connectivity patterns between users and items. The encoder of contextual features is defined as $h(\cdot) : \mathbb{R}^{d^c} \to \mathbb{R}^{d'}$, where d' is the dimension of encoded contextual embeddings. Depending on the feature type (i.e., categorical or numerical), the contextual embedding process can be formulated as look-up tables for categorical features or a deep neural network for numerical features.

To predict the probability that user i interacts with item j, the input $\mathbf{z}_{ij} \in \mathbb{R}^{2d+d'}$ to a CTR model is:

$$\mathbf{z}_i = f(x_i),\ \mathbf{z}_j = f(x_j),\ \mathbf{z}^c_{ij} = h(\mathbf{c}_{ij}), \tag{1}$$

$$\mathbf{z}_{ij} = \left[\mathbf{z}_i \parallel \mathbf{z}_j \parallel \mathbf{z}^c_{ij}\right], \tag{2}$$

where $\parallel$ refers to the concatenation operation, $\mathbf{z}_i$ and $\mathbf{z}_j$ refer to the latent ID embeddings of user i and item j generated by the encoder respectively, and $\mathbf{z}^c_{ij}$ is the encoded contextual features. With the encoded embeddings, each CTR model adopts its corresponding rating function $r(\cdot) : \mathbb{R}^{2d+d'} \to \mathbb{R}$ that models feature interaction to predict the likelihood of interaction. Specifically, let $p_{ij} = r(\mathbf{z}_{ij})$, where $p_{ij} \in [0, 1]$ represents how likely user i would interact with item j. The CTR model is trained with the binary cross-entropy loss with weight regularization:

$$\mathcal{L} = -\frac{1}{N} \sum_{\{i,j\} \in T_r} y_{ij} \log(p_{ij}) + (1 - y_{ij}) \log(1 - p_{ij}) + \lambda \sum_l \|\mathbf{w}^{(l)}\|^2, \tag{3}$$

where T_r denotes the training set, y_{ij} refers to the binary label (1 for

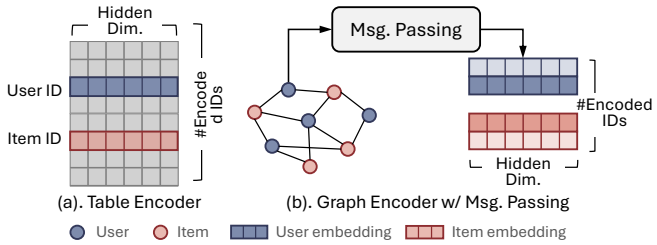

Figure 1: Encoders to encode user and item IDs.

positive and 0 for negative), λ is the L_2 regularization coefficient, and l is the layer number of the model. Importantly, these embeddings are structurally blind to the rich connectivity patterns between users and items.

Message Passing. In stark contrast stands the principle of Message Passing (MP), a cornerstone of GNNs designed explicitly for learning from such *relational* data [6, 14, 28]. To formulate the representation exchange in recommendation, let the interaction matrix be $R \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$, where $r_{uv} = 1$ represents an observed valid interaction between user u and item v, and 0 otherwise. To extract collaborative signals, the interaction matrix R is abstracted to a bipartite graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ is the set of nodes and $\mathcal{E} = \{(u,v)|u \in \mathcal{U}, v \in \mathcal{I}, r_{uv} = 1\}$ is the set of edges. A linear layer that updates node features using MP has the following update rule:

$$\mathbf{h}_i' = \sigma\left(W\left(\mathbf{h}_i + \text{Agg}\left(\{\mathbf{h}_j : j \in \mathcal{N}_i\}\right)\right)\right), \tag{4}$$

where $\mathbf{h}_i'$ is node i's updated embedding from $\mathbf{h}_i$, $\mathcal{N}_i$ is the set of neighbors of node i, W is a shared weight matrix, σ is a Lipschitz continuous activation function with Lipschitz constant L_σ, and $\text{Agg}(\cdot)$ is the averaging function. We depict this embedding process in Figure 1 (b). Notably, the form of connectivity we emphasize is fundamentally orthogonal and distinct from that considered in prior graph-based CTR methods [15, 20, 36], which focus on feature-level relationships rather than on user-item interactions as in our study.

4 Operationalizing Connectivity for Stability

4.1 The Lipschitz Constant: A Measure of Model Stability

To clarify the theoretical foundation of our argument, we first examine a key property of deep models, namely stability, quantified by the Lipschitz constant. The Lipschitz constant is a critical concept for understanding and controlling the stability of deep models. A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is said to be Lipschitz continuous on an input set $\mathcal{X} \subseteq \mathbb{R}^n$ if there exists a bound $K \geq 0$ such that for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, f satisfies:

$$\|f(\mathbf{x}) - f(\mathbf{y})\| \leq K\|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathcal{X}. \tag{5}$$

The smallest possible K in Equation 5 is defined as the Lipschitz constant of f, denoted as $\text{Lip}(f)$:

$$\text{Lip}(f) = \sup_{\mathbf{x},\mathbf{y}\in\mathcal{X},\mathbf{x}\neq\mathbf{y}} \frac{\|f(\mathbf{x}) - f(\mathbf{y})\|}{\|\mathbf{x} - \mathbf{y}\|}. \tag{6}$$

In this context, f is referred to as a K-Lipschitz function, where the Lipschitz constant K quantifies the maximum change in the function's output caused by a unit-norm perturbation in its input. This constant serves as a crucial indicator of a deep model's stability wrt its inputs: a model with a smaller Lipschitz constant indicates lower sensitivity to input variations and exhibits greater stability.

4.2 Connectivity-Driven Stability in Message Passing

Grounding CTR prediction in the structural reality of user-item connectivity is theoretically supported to yield inherently more stable models than those built on isolated features alone. In particular, we examine how incorporating connectivity through MP enhances stability at the level of representation learning, leading to more generalizable embeddings.

> THEOREM 4.1. *For any node $i \in \mathcal{V}$ and its node feature $\mathbf{h}_i$, consider the mapping function with the updating rule demonstrated in Equation 4. Then, the Lipschitz constant L of the mapping $f : \mathbf{h} \mapsto \mathbf{h}'$, where $\mathbf{h}, \mathbf{h}' \in \mathbb{R}^d$ are the collection of all node features in the graph before and after the mapping, respectively, satisfies:*
>
> $$L \leq L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}}, \tag{7}$$
>
> *where $\|W\|_F$ is the Frobenius norm of the weight matrix W, k_i is the degree of node i, L_σ is the Lipschitz constant of a Lipschitz continuous activation function $\sigma(\cdot)$ (e.g., ReLU), and d is the dimension of node feature.*

Insight. Within the MP, an increased node degree k_i for node i reduces the Lipschitz constant of the mapping function, thereby stabilizing the model output wrt input variations. Then, stable representations make the model less likely to be erratically swayed by the sparse or foreign signals from individual, less-represented users. This stability, born from connectivity, helps keep predictive performance more evenly distributed rather than being captured by outliers or noise. For full proof, please refer to Appendix B.

4.3 Operationalizing Connectivity in CTR Models

The user-item connectivity can be grounded in practical, well-established methods. To demonstrate the concrete benefits of leveraging this structure, we employ MP via a minimal, widely adopted approach in recommendations. Rather than introducing unnecessary complexity, we adopt a linear MP module [10] as the operator for a controlled intervention. This minimal MP operator omits the non-linearities and weight matrices typical of more complex GNNs (see Equation 4), letting us isolate the effect of graph connectivity from the feature-interaction architecture of downstream CTR models.

Specifically, let $f_g(\cdot,\cdot) : \mathcal{G} \times x \to \mathbb{R}^d$ be the graph encoder. For user i and item j, $f_g(\cdot,\cdot)$ conducts MP in each layer to propagate and aggregate information from the neighborhood. The graph-encoded

Table 1: The metadata of the curated datasets. k is the least number of interactions per user and item. Density = (#Inters / (#Users × #Items)) ×100%.

k	Users	Int./User	Items	Int./Item	Inters	Density%
ML-1M (*dense*)						
3	6K	165.6	3.5K	285.5	1M	4.72
6	6K	165.5	3.4K	296.0	1M	4.90
10	6K	165.3	3.3K	306.3	1M	5.07
Yelp2022 (*sparse*)						
3	529K	9.8	144K	36.2	5.2M	0.01
6	215K	17.9	96K	40.3	3.9M	0.02
10	99K	27.9	56K	48.9	2.8M	0.05

embedding for node i (user or item) is obtained with the following:

$$\mathbf{z}_i = f_g(\mathcal{G}, x_i) = \sum_{l=0}^{L} a_l \mathbf{z}_i^{(l)}, \tag{8}$$

$$\mathbf{z}_i^{(l)} = \sum_{v \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_j|}} \mathbf{z}_j^{(l-1)}, \mathbf{z}_j^{(0)} = f(x_j), \tag{9}$$

where $\mathbf{z}_i^{(l)}$ is the embedding for node i in layer l, $\mathcal{N}_i$ is the set of neighbors for node i in $\mathcal{G}$, a_l is the readout coefficient for each layer-l's embeddings, and L is the number of MP layers. With the obtained graph-based user and item ID embeddings, we construct the input features following Equation 2 and feed them into any CTR method (e.g., DCN [33]) to predict the interaction probabilities. This formulation serves as a practical demonstration of how connectivity can be introduced as an analytical intervention while keeping the downstream CTR model unchanged.

5 Experiments

We formulate the minimal MP as a controlled probe for studying user-item connectivity. Following the stability analysis, our experiments ask whether the interaction graph is useful in practice, under what density conditions it yields reliable gains, and how those gains are distributed across users. We therefore apply the same MP intervention to several strong CTR backbones, evaluate it on datasets and k-core variants with different interaction densities, and compare against the corresponding non-MP baselines.

Moreover, Theorem 4.1 suggests that the stability effect of MP depends on node degree, implying varied impacts across users with different activity levels. To investigate this distributional dimension beyond aggregate performance metrics, we introduce the Marginal Cost of Improvement (MCI) metric to quantify how predictive gains are allocated across user groups. Together, these experiments clarify when connectivity-driven stability translates into consistent performance improvements and how those improvements are distributed within the population.

5.1 Setup

Datasets to Probe Density Effects. Our analysis employs two public recommendation benchmarks with contrasting *global* interaction densities: MovieLens-1M [7], a dense dataset with roughly 1M ratings from 6K users on 4K movies, and Yelp2022 [12], a much sparser dataset containing user reviews, item information, and user-item interactions. This contrast allows us to test whether MP's effectiveness depends on the amount of available connectivity. For finer-grained analysis of *local* density, we apply k-core filtering [8], which retains users and items with at least k interactions; larger k corresponds to a locally denser interaction graph. To analyze supervision reliability, we use WS to denote whether middle-rating samples are retained (✓) or removed (✗). For all datasets, ratings are converted into binary labels through thresholding, and standard 80/10/10 train/validation/test splits are used with results averaged over five random seeds.

Table 2: Relative AUC change (Δ%) from TS versus Base. Cell colors encode significant improvement or decrement; grey denotes no significant change. Mean/#Gains aggregates over the three k settings; Overall rows aggregate within each dataset.

Dataset	Model	$k=3$	$k=6$	$k=10$	Mean/#Gains
ML-1M (dense)	DCNv2	+0.16	+0.18	+0.16	+0.17/3
	EulerNet	+0.30	+0.15	+0.28	+0.24/3
	FinalMLP	+0.03	+0.09	+0.15	+0.09/2
	GDCNP	+0.32	+0.28	+0.33	+0.31/3
	AFN	−0.02	+0.10	+0.16	+0.08/2
	AutoInt	+0.08	+0.12	+0.07	+0.09/3
	Overall	+0.15	+0.15	+0.19	+0.16/16
Yelp2022 (sparse)	DCNv2	−1.66	−1.61	−0.01	−1.09/0
	EulerNet	+0.25	+0.52	+0.15	+0.31/3
	FinalMLP	+0.06	+0.12	+0.54	+0.24/3
	GDCNP	+0.31	+0.83	+0.63	+0.59/3
	AFN	−0.13	+0.55	+0.29	+0.24/2
	AutoInt	−0.60	−1.15	−2.16	−1.30/0
	Overall	−0.30	−0.12	−0.09	−0.17/11

Baselines. To ensure that our analysis is not tied to a single CTR architecture, we evaluate MP across six representative feature-interaction models: DCNv2 [33], AutoInt [26], EulerNet [27], AFN [1], FinalMLP [19], and GDCNP [31]. These models share a common contribution goal: improving CTR prediction by designing stronger mechanisms for modeling interactions among encoded features, while still treating user and item IDs primarily as independently learned feature embeddings. They cover several mechanisms adopted in prior CTR research, including explicit cross networks and deep components (DCNv2), self-attention over feature fields (AutoInt), hyperspace or transformed feature interactions (EulerNet and AFN), lightweight MLP-based interaction learning (FinalMLP), and graph-enhanced cross networks over feature relations (GDCNP). Evaluating the same MP intervention across these mechanisms lets us test whether the role of user-item connectivity is robust to the model's feature-interaction design, rather than being an artifact of one particular baseline. Notably, we left out prior graph-based CTR methods [15, 20, 36] from the baseline models, as they focus on feature-level relationships rather than user-item interactions,

Table 3: AUC comparison among Base, fine-tuning with MP (FT), and training from scratch with MP (TS). FT and TS cells report AUC with relative change (Δ%) from Base. WS indicates whether middle-rating samples are retained (✓) or removed (✗) during evaluation; training retains them. Cell colors encode significant improvement or decrement; grey denotes no significant change.

Dataset	WS	k	DCNv2			AutoInt			FinalMLP		
			Base	FT(Δ%)	TS(Δ%)	Base	FT(Δ%)	TS(Δ%)	Base	FT(Δ%)	TS(Δ%)
ML-1M	✓	3	82.20	82.51 (0.38)	82.33 (0.16)	82.11	82.28 (0.22)	82.17 (0.08)	82.18	82.30 (0.14)	82.21 (0.03)
		6	82.20	82.53 (0.40)	82.34 (0.18)	82.14	82.32 (0.22)	82.24 (0.12)	82.15	82.31 (0.19)	82.22 (0.09)
		10	82.11	82.25 (0.17)	82.24 (0.16)	81.99	82.20 (0.26)	82.05 (0.07)	82.04	82.15 (0.13)	82.17 (0.15)
	✗	3	89.57	89.87 (0.33)	89.69 (0.14)	89.47	89.66 (0.22)	89.56 (0.11)	89.50	89.61 (0.12)	89.55 (0.06)
		6	89.56	89.84 (0.31)	89.67 (0.12)	89.50	89.70 (0.22)	89.59 (0.11)	89.44	89.62 (0.20)	89.56 (0.14)
		10	89.41	89.60 (0.21)	89.58 (0.19)	89.28	89.47 (0.21)	89.37 (0.10)	89.33	89.44 (0.12)	89.44 (0.12)
Yelp2022	✓	3	81.65	80.49 (-1.42)	80.30 (-1.66)	82.06	81.84 (-0.26)	81.57 (-0.60)	79.82	79.52 (-0.37)	79.87 (0.06)
		6	80.48	79.52 (-1.19)	79.19 (-1.61)	79.88	79.72 (-0.20)	78.96 (-1.15)	78.18	77.91 (-0.35)	78.27 (0.12)
		10	77.22	77.61 (0.51)	77.21 (-0.01)	78.09	78.13 (0.05)	76.41 (-2.16)	76.56	76.73 (0.22)	76.97 (0.54)
	✗	3	86.35	85.93 (-0.48)	85.72 (-0.72)	86.95	86.76 (-0.23)	86.56 (-0.45)	84.63	84.34 (-0.34)	84.82 (0.23)
		6	85.73	85.09 (-0.75)	84.55 (-1.38)	85.08	84.92 (-0.20)	84.30 (-0.92)	83.12	82.77 (-0.43)	83.40 (0.33)
		10	82.30	83.11 (0.98)	82.70 (0.49)	83.39	83.48 (0.10)	81.88 (-1.82)	81.57	81.85 (0.34)	82.22 (0.79)

the form of connectivity we emphasize.

Training and Evaluation Configurations. Models are trained with binary cross-entropy (Equation 3) optimized by the AdamW optimizer, employing rigorous hyperparameter tuning for optimal AUC on validation sets. The primary evaluation metric discussed is AUC, while our Marginal Cost of Improvement (MCI) metric (§5.4, Equation 10) assesses how predictive gains are distributed across user groups. Logloss is also reported in Appendix A.1 to provide a more comprehensive view of model performance.

MP Integration Strategies. To understand practical implementation pathways, we analyze three simple types of MP integration: *(i)* Base (no MP), serving as the baseline; *(ii)* Fine-Tuning (FT), where MP was integrated into pre-trained Base models; and *(iii)* Training from Scratch (TS), where models incorporated MP from their initial state. This tiered approach is essential for evaluating the efficiency and effectiveness of different adoption strategies for MP.

5.2 High Connectivity Density is the Key Enabler for Connectivity's Benefits

Our systematic comparisons are summarized in Table 2: each cell reports the relative AUC change from training with MP (TS) compared with the baseline feature-centric CTR model (Base) across datasets with varying interaction densities. Overall, MP yields CTR gains in 27 out of 36 test cases. However, both the magnitude and consistency of these improvements differ substantially between datasets. On the globally dense ML-1M dataset, 16 of 18 experiments show gains, with uniformly positive average changes across the three k settings and very few negative outcomes. In contrast, on the globally sparse Yelp2022 dataset, gains occur in only 11 of 18 cases and are more sensitive to the underlying model architecture.

This contrast reflects the structural role of interaction density. In denser graphs, richer connectivity provides more reliable relational signals, enabling MP to propagate and aggregate information more effectively. The resulting representations are better supported by a larger number of neighboring interactions and thus more stable and informative. In sparse settings, however, limited connectivity constrains information flow, weakening the stabilizing effect of MP and increasing sensitivity to idiosyncratic or noisy interactions.

Table 2 also indicates that MP's effectiveness is governed primarily by the global connectivity of the interaction graph rather than by local adjustments introduced through k-core filtering. We do not observe a consistent pattern in which increasing k systematically enhances MP's performance.

We further verify that this density-dependent pattern is not merely an artifact of how weakly supervised middle-rating samples are handled. In Appendix 8, we compare settings that retain or remove such samples during training and evaluation while matching dataset density as closely as possible. The same qualitative pattern persists: MP yields stable gains on dense ML-1M across supervision settings, whereas its effect on sparse Yelp2022 remains model-dependent. This suggests that supervision filtering alone does not explain the observed benefits; sufficient connectivity remains the key condition for reliable MP improvement.

This observed dependence of MP's effectiveness on connectivity density also carries practical implications for recommender system research and deployment. In many real-world scenarios, sparse

Table 4: The AUC performance on ML-1M (6-core) and Yelp2022 (10-core) across users with varying activity levels. Base refers to the original CTR model, TS refers to training from scratch with MP, and FT refers to fine-tuning from the original model with MP. *Low, Mid, High* refer to low, medium, and high user activity, respectively. Var is the variance of the performance among the three groups. Under-served group with the lowest performance level is bolded.

ML-1M Dense Data									
Model	DCNv2			AutoInt			FinalMLP		
Variant	Base	FT(Δ%)	TS(Δ%)	Base	FT(Δ%)	TS(Δ%)	Base	FT(Δ%)	TS(Δ%)
Overall ↑	82.20	82.53 (0.40)	82.34 (0.18)	82.14	82.32 (0.22)	82.24 (0.12)	82.15	82.31 (0.19)	82.22 (0.09)
Low ↑	**76.51**	77.02 (0.67)	77.00 (0.63)	**76.61**	76.97 (0.48)	76.76 (0.19)	**76.47**	76.60 (0.18)	76.66 (0.25)
Mid ↑	79.91	80.34 (0.53)	80.12 (0.25)	79.87	80.10 (0.29)	79.97 (0.12)	79.98	80.16 (0.22)	80.03 (0.06)
High ↑	83.24	83.5 (0.31)	83.32 (0.10)	83.15	83.29 (0.17)	83.24 (0.12)	83.15	83.29 (0.17)	83.22 (0.08)
Var ↓	11.30	10.48 (-7.28)	10.01 (-11.43)	10.69	9.97 (-6.73)	10.52 (-1.58)	11.17	11.2 (0.34)	10.76 (-3.62)
Yelp2022 Sparse Data									
Overall ↑	77.22	77.61 (0.51)	77.21 (-0.01)	78.09	78.13 (0.05)	76.41 (-2.16)	76.56	76.73 (0.22)	76.97 (0.54)
Low ↑	79.34	79.25 (-0.11)	79.02 (-0.41)	79.70	79.61 (-0.10)	76.79 (-3.65)	76.84	76.89 (0.07)	77.48 (0.84)
Mid ↑	78.52	78.63 (0.13)	78.27 (-0.32)	79.14	79.11 (-0.04)	76.87 (-2.86)	77.02	77.18 (0.21)	77.55 (0.69)
High ↑	**76.14**	76.79 (0.85)	76.39 (0.32)	**77.23**	77.36 (0.17)	76.21 (-1.32)	**76.30**	76.49 (0.26)	76.60 (0.39)
Var ↓	2.77	1.64 (-40.61)	1.84 (-33.60)	1.67	1.40 (-16.51)	0.13 (-92.17)	0.14	0.12 (-14.93)	0.28 (102.12)

user–item interaction matrices pose a well-recognized challenge for recommendation models [2, 25]. **These findings show that user-item connectivity becomes a useful resource for CTR prediction only when the interaction graph is sufficiently informative**, allowing the stability advantages theorized earlier to translate into consistent empirical gains.

5.3 Practical Pathways: Integrating Connectivity via Simple Fine-Tuning

Integrating connectivity-aware mechanisms into existing CTR models presents practical challenges. A resource-efficient alternative is to fine-tune (FT) feature-centric models with MP-enhanced embeddings, rather than training new MP-integrated systems from scratch (TS). We systematically assess the relative effectiveness of these integration strategies in Table 3. Results show that the FT approach consistently delivers better or similar relative performance improvements in comparison to TS: across all datasets and model architectures, FT achieved an average AUC improvement of 0.22% over TS. Moreover, FT mitigates the negative effects of MP in sparser datasets more effectively than training from scratch TS.

These findings indicate that the representational capacity learned by well-trained feature-centric models need not be discarded; instead, relational information from MP can be incorporated to enhance existing well-trained architectures. FT not only delivers more consistent performance gains but also improves computational efficiency, as it reuses learned parameters and typically converges faster than training a new MP-integrated model from scratch (TS).

Notably, in the dense ML-1M dataset, the gains from FT remained consistent regardless of whether weakly supervised samples were included ✓ or excluded ✗ during evaluation. This pattern suggests that the improvement is not concentrated on already well-supported or "easier" instances where supervision signals are relatively more reliable (i.e., larger rating gaps between positive and negative items), but extends to samples receiving weaker supervision signals as well. Such breadth of enhancement indicates a more balanced allocation of predictive performance improvement, benefiting both strongly and weakly supported instances rather than amplifying advantages only where signals are already reliable. Altogether, the results suggest a practical deployment path: **Connectivity-aware embeddings can often be introduced through fine-tuning while keeping the underlying CTR architecture unchanged, avoiding the need to train MP-enhanced models from scratch**.

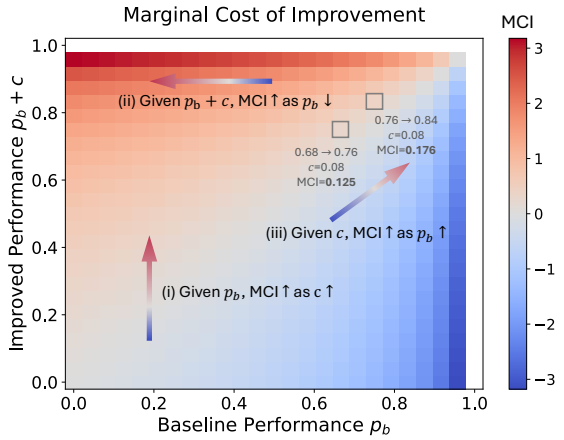

Figure 2: MCI for different performance levels. The MCI is defined as $MCI(p_b, c) = \log((1 - p_b)/(1 - p_b - c))$, where p_b is the baseline performance, c is the absolute performance change, and $p_b + c$ is the new performance.

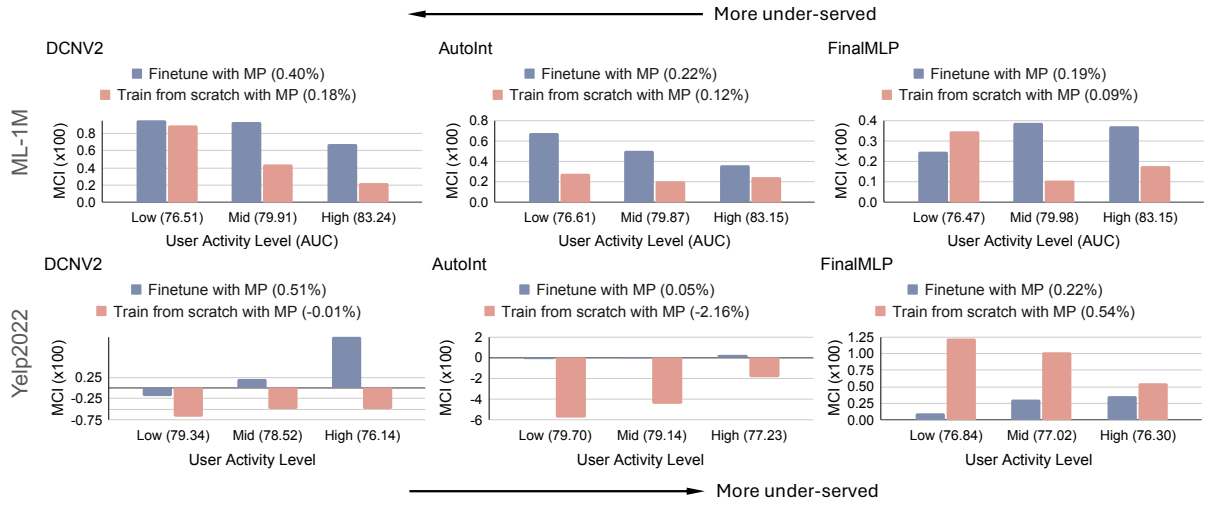

Figure 3: MCI-evaluated cost for AUC improvement (MCI > 0) and downgrade (MCI < 0) across users with different activity levels. The under-served groups in ML-1M and *Yelp2022* are low-activity and *high-activity* users, respectively.

5.4 Redistributing Predictive Gains Toward Under-Served Users

We next examine how connectivity changes the distribution of predictive gains across users. Since Theorem 4.1 shows that MP's stability effect depends on node degree, we stratify users by activity level, which corresponds to their degree in the interaction graph. This lets us compare how much predictive improvement is directed toward different activity-defined user groups.

We define under-served users as those whose predictive performance is relatively poorer compared to their peers, and identify the under-served group separately in each dataset based on group-level predictive performance. Table 4 shows that low-activity users are more under-served in the dense ML-1M dataset, whereas high-activity users are more under-served in the sparse Yelp2022 dataset. Users who are under-served often include niche-content enthusiasts and infrequent users whose behaviors deviate from dominant interaction patterns. Feature-centric models may struggle to capture the relational context underlying these distinctive behaviors, leading to weaker predictive support. The reduced performance variance across groups with MP-integrated connectivity indicates a more balanced distribution of predictive performance.

To characterize how MP reshapes the distribution of predictive improvement across users, inspired by classical *marginal utility* [21, 29], we define the Marginal Cost of Improvement (MCI) to quantify the "effort" a model expends to improve performance for a given segment. Formally, let $p_b \in (0,1)$ be the baseline AUC and c the absolute AUC change (so $p_b + c \in (0,1)$). We define:

$$\mathrm{MCI}(p_b, c) \;=\; \log\!\Big(\frac{1-p_b}{1-p_b-c}\Big), \tag{10}$$

where $c > 0$ means MCI > 0 (improvement), and $c < 0$ means MCI < 0 (decline).

Importantly, MCI meets three intuitive criteria based on marginal utility, making it a suitable measure of the effort a model expends: *(i)* Under the same baseline performance (p_b), the larger the absolute change in performance (c), the higher the cost (MCI); *(ii)* Under the same ultimate performance ($p_b + c$), the lower the baseline performance, the higher the cost; *(iii)* Under the same absolute change in performance, the lower the baseline performance, the higher the cost required or utility lost (i.e., |MCI| is larger). A proof of MCI's properties is provided in Appendix C, and Figure 2 illustrates the intuition of the three criteria.

Figure 3 examines the MCI for each group to evaluate the amount of effort the models put in each group, revealing: *(i)* In the dense ML-1M data, MP-enhanced models, via fine-tuning (FT) or training from scratch (TS), allocate the most effort (highest MCI) into improving performance for the under-served group (low-activity users); *(ii)* Even in the sparse Yelp2022 dataset, MP mainly benefits the under-served group (high-activity users) by reducing performance imbalance across groups; even when performance drops, these users lose the least MCI. These findings collectively show that **MP often shifts more effective improvement toward user groups that are initially less well served by feature-centric CTR models**. This redistribution indicates that connectivity-aware representations can change how predictive gains are allocated across activity-defined user segments, rather than confining improvements to already well-supported users.

6 Conclusion and Limitations

This work studies user–item connectivity as an object of analysis for CTR prediction. Using a minimal MP operator as a controlled intervention, we isolate the effect of the graph connectivity from the feature-interaction architecture of downstream CTR models. Theoretically, we provide a stability-based account showing that MP smooths representations over the interaction graph and reduces sensitivity to input perturbations. Empirically, we show that connectivity is not uniformly beneficial as dense interaction graphs yield more consistent gains, while sparse graphs produce less consistent outcomes. We further show that connectivity-aware representations can be incorporated efficiently through fine-tuning existing feature-centric models. Additionally, we find that MP often shifts

more effective improvement toward initially under-served users.

These findings suggest that user–item connectivity should be treated as a measurable data resource in CTR prediction, not merely as an auxiliary modeling signal. When the interaction graph contains sufficient reliable structure, connectivity-aware representations can improve predictive performance while also changing how gains are distributed across user groups.

This study has limitations that point to future research directions. Our empirical analysis focuses on two public datasets and one deliberately minimal MP integration design, so broader validation across additional domains, industrial-scale systems, and alternative connectivity mechanisms remains important. Finally, while MCI provides a useful lens on group-level redistribution, it does not by itself establish broader fairness claims. Future work should examine finer-grained user outcomes, and the long-term dynamics of connectivity-aware recommendation.

Supplementary Material

A Further Analysis

A.1 Logloss Results

We provide the Logloss results in Table 5 and 6, corresponding to the AUC comparisons in Table 2 and 3. Note that, unlike AUC, the lower the Logloss, the more reliable the output.

The results in Table 5 generally align with those in Table 2. In the dense dataset ML-1M, incorporating MP under the TS configuration not only improves the AUC performance but also enhances prediction confidence. Similar to the observations from Table 2, in the sparse dataset Yelp2022, we find the benefits of MP in enhancing the reliability of the predictions is model-dependent. For instance, in DCNV2 [33], MP consistently improves prediction reliability enhancement, whereas in EulerNet [27], MP proves beneficial only when the local dataset is low. However, for AutoInt [26], incorporating MP negatively impacts performance across all k values.

The results in Table 6 also align with those in Table 3. Comparing with TS, incorporating MP under FT is more likely to enhance model prediction confidence, as reflected in lower Logloss values. Even when MP decreases confidence, FT mitigates the negative effect brought by MP more effectively than TS. Notably, we observe that prediction confidence in strongly supervised samples is lower than the overall confidence, suggesting that these samples exhibit more general patterns, making them easier for the model to distinguish. Consequently, improving confidence in predicting strongly supervised samples is more challenging, as their confidence levels are already high, whereas noisy, weakly supervised samples offer more room for confidence improvement.

A.2 Impact of Supervision Reliability on MP in CTR Prediction

We vary the inclusivity of weakly-supervised samples during both training and testing to further investigate how supervision reliability affects the effectiveness of integrating MP into CTR prediction. To control for the influence of connectivity density, we group curated datasets into pairs with **similar densities**. By comparing results within each pair, we can isolate and analyze the impact of supervision reliability during training while minimizing the effects of the confounding factor of dataset density. The metadata for the paired curated datasets is provided in Table 7.

The results in Table 8 indicate that excluding weakly supervised samples during training improves the performance of strongly supervised samples. In the dense dataset ML-1M, MP consistently

Table 5: Relative Logloss change (Δ%) from TS versus Base, matching the organization of Table 2. Lower Logloss is better, so negative values indicate improved prediction confidence. Mean/#Gains aggregates over the three k settings; Overall rows aggregate within each dataset.

Dataset	Model	$k=3$	$k=6$	$k=10$	Mean/#Gains
ML-1M (dense)	DCNv2	−0.60	−0.51	−0.25	−0.45/3
	EulerNet	−0.21	−0.64	−1.08	−0.64/3
	FinalMLP	−0.03	−0.23	−0.09	−0.12/3
	GDCNP	−0.27	−0.39	−0.58	−0.41/3
	AFN	+0.02	−0.16	−0.29	−0.14/2
	AutoInt	−0.38	−0.24	−0.33	−0.32/3
	Overall	−0.25	−0.36	−0.44	−0.35/17
Yelp2022 (sparse)	DCNv2	−1.55	−0.29	−9.31	−3.72/3
	EulerNet	−1.29	+0.81	+0.97	+0.16/1
	FinalMLP	−1.27	−0.11	−0.48	−0.62/3
	GDCNP	−0.69	−2.10	−0.97	−1.25/3
	AFN	−1.19	−2.44	+1.13	−0.83/2
	AutoInt	+1.25	+22.41	+8.80	+10.82/0
	Overall	−0.62	+3.05	+0.02	+0.82/12

Table 6: Relative Logloss change (Δ%) for the same settings as Table 3. Lower Logloss is better, so negative values indicate improved prediction confidence. The layout mirrors Table 3, but only reports FT and TS deltas from Base to avoid repeating all absolute Logloss values.

Dataset	WS	k	DCNv2		AutoInt		FinalMLP	
			FT	TS	FT	TS	FT	TS
ML-1M	✓	3	-1.72	0.52	-0.54	1.47	-0.81	-0.31
		6	-2.92	-1.73	-0.37	-1.84	-0.06	0.57
		10	-0.87	0.31	2.22	3.90	-1.51	0.04
	✗	3	-1.14	-0.60	-0.56	-0.38	-0.35	-0.03
		6	-1.24	-0.51	-0.64	-0.24	-0.45	-0.23
		10	-0.77	-0.25	-0.86	-0.33	-0.32	-0.09
Yelp2022	✓	3	4.68	6.00	1.38	3.27	-0.89	-0.75
		6	-10.11	-9.86	-1.97	42.58	0.33	0.23
		10	-4.84	-8.48	0.87	21.33	-0.07	-1.73
	✗	3	-2.49	-1.55	0.33	1.25	-0.95	-1.27
		6	-2.38	-0.29	-0.55	22.41	-0.11	-0.11
		10	-8.64	-9.31	-0.37	8.80	-0.63	-0.48

Table 7: Paired dataset settings with similar density. *TrWS* denotes whether weakly supervised middle-rating samples are retained (✓) or removed (✗) during training.

(a) ML-1M dense pairs

Pair	*TrWS*	k	Users	Items	Density%
M1	✓	1	6.0K	3.7K	4.47
M1	✗	19	5.7K	2.9K	4.44
M2	✓	3	6.0K	3.5K	4.72
M2	✗	22	5.4K	2.8K	4.74

(b) Yelp2022 sparse pairs

Pair	*TrWS*	k	Users	Items	Density%
Y1	✓	3	0.5M	0.14M	0.01
Y1	✗	3	0.5M	0.14M	0.01
Y2	✓	6	0.2M	96K	0.02
Y2	✗	6	0.2M	89K	0.02

enhances performance regardless of the inclusion of weakly supervised samples, with relative improvements remaining stable across both weakly and strongly supervised samples. However, in sparse datasets, training with weakly supervised samples affects the integration of MP—when MP leads to performance gains, strongly supervised samples experience higher relative improvement, while in cases of performance degradation, the negative effects on strongly supervised samples are mitigated. Furthermore, we again observe that the effectiveness of MP in the sparse Yelp2022 dataset is highly model-dependent, emphasizing the need for careful consideration when integrating MP in sparse datasets.

B Proof of Theorem 4.1

Theorem 4.1 *For any node $i \in \mathcal{V}$ and its node feature $\mathbf{h}_i$, consider the mapping function with the updating rule demonstrated in Equation 4. Then, the Lipschitz constant L of the mapping $f : \mathbf{h} \mapsto \mathbf{h}'$, where $\mathbf{h}, \mathbf{h}' \in \mathbb{R}^d$ are the collection of all node features in the graph before and after the mapping respectively, satisfies:*

$$L \leq L_\sigma \|W\|_F \sqrt{1 + \frac{1}{k_i}},$$

where $\|W\|_F$ is the Frobenius norm of the weight matrix W, k_i denotes the degree of node i, L_σ is the Lipschitz constant of a Lipschitz continuous activation function $\sigma(\cdot)$, and d is the dimension of the node feature.

Proof. We aim to estimate the Lipschitz constant L of the mapping $f : \mathbf{h} \mapsto \mathbf{h}'$, where $\mathbf{h}$ and $\mathbf{h}'$ denote the collection of all node features in the graph before and after the mapping, respectively. The Lipschitz constant measures how much the output of the function can change with respect to changes in the input, and can be estimated by analyzing the Jacobian matrix $\mathbf{J}$ of the mapping.

Step 1: Define the Jacobian of the Mapping

Consider the mapping for node i:

$$\mathbf{h}'_i = \sigma\left(W\left(\mathbf{h}_i + \frac{1}{k_i}\sum_{j\in\mathcal{N}(i)} \mathbf{h}_j\right)\right) = \sigma\left(W\mathbf{s}_i\right), \tag{11}$$

Table 8: Relative AUC change (Δ%) from training with MP (TS) versus the original CTR model (Base) under different weak-supervision settings. *TrWS* and *EvalWS* indicate whether weakly supervised middle-rating samples are retained (✓) or removed (✗) during training and evaluation, respectively. Dataset pairs are matched to similar density to isolate the effect of supervision reliability. Pair IDs refer to the dataset settings in Table 7. Cell colors encode significant improvement or decrement; grey denotes no significant change.

Pair	*TrWS*	*EvalWS*	Model			
			DCNv2	FinalMLP	GDCNP	AutoInt
M1	✓	✓	+0.14	+0.03	+0.35	+0.04
M1	✓	✗	+0.11	+0.04	+0.35	+0.04
M1	✗	✗	+0.21	+0.11	+0.33	+0.06
M2	✓	✓	+0.16	+0.03	+0.32	+0.08
M2	✓	✗	+0.14	+0.06	+0.35	+0.11
M2	✗	✗	+0.08	+0.16	+0.31	+0.08
Y1	✓	✓	−1.66	+0.06	+0.31	−0.60
Y1	✓	✗	−0.72	+0.23	+0.37	−0.45
Y1	✗	✗	−1.33	−0.17	+0.47	−0.14
Y2	✓	✓	−1.61	+0.12	+0.83	−1.15
Y2	✓	✗	−1.38	+0.33	+0.86	−0.92
Y2	✗	✗	−0.49	−0.09	+0.68	+0.59

where $\mathbf{s}_i = \mathbf{h}_i + \mathbf{h}_{\text{agg}}$ and $\mathbf{h}_{\text{agg}} = \frac{1}{k_i}\sum_{j\in\mathcal{N}(i)} \mathbf{h}_j$. The Jacobian matrix $\mathbf{J}$ of this mapping is the matrix of partial derivatives of $\mathbf{h}'_i$ wrt all node features $\mathbf{h}_j$ for $j \in \mathcal{V}$:

$$\mathbf{J} = \frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}^\top}, \tag{12}$$

where $\mathbf{h}'_i \in \mathbb{R}^d$ is the updated feature vector of node i, and $\mathbf{h} \in \mathbb{R}^{|\mathcal{N}|d}$ denotes the concatenation of all node features in the graph.

Step 2: Express the Jacobian matrix

The Jacobian matrix can be partitioned into blocks corresponding to derivatives for $\mathbf{h}_i$ and $\mathbf{h}_j, \forall j \in \mathcal{N}(i)$:

(i) For the partial derivative of $\mathbf{h}'_i$ wrt $\mathbf{h}_i$, we have

$$\frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_i^\top} = D_i W, \tag{13}$$

where $D_i = \text{diag}\left(\sigma'\left(W\mathbf{s}_i\right)\right)$ is a diagonal matrix containing the derivatives of the activation function applied element-wise, and $\sigma'(\cdot)$ is the elementwise derivative of the activation function $\sigma(\cdot)$.

(ii) For the partial derivative of $\mathbf{h}'_i$ wrt $\mathbf{h}_j, \forall j \in \mathcal{N}(i)$:

$$\frac{\partial \mathbf{h}'_i}{\partial \mathbf{h}_j^\top} = \begin{cases} D_i W \cdot \frac{1}{k_i}, & \text{if } j \in \mathcal{N}(i), \\ 0, & \text{otherwise.} \end{cases} \tag{14}$$

This distinction of the two cases arises because $\mathbf{h}'_i$ depends only on the features of node i and its neighbors $j \in \mathcal{N}(i)$.

Step 3: Bound the Partial Derivatives

To estimate the Lipschitz constant, we need to estimate the spectral norm of the Jacobian $\|\mathbf{J}\|_2$. Since the spectral norm is less than or equal to the Frobenius norm [11], we seek to compute the Frobenius norm of $\mathbf{J}$:

$$\|\mathbf{J}\|_F = \sqrt{\left\|\frac{\partial \mathbf{h}_i'}{\partial \mathbf{h}_i^\top}\right\|_F^2 + \sum_{j\in\mathcal{N}(i)}\left\|\frac{\partial \mathbf{h}_i'}{\partial \mathbf{h}_j^\top}\right\|_F^2}. \tag{15}$$

We now simplify the expression separately. We simplify the first Frobenius norm as:

$$\left\|\frac{\partial h_i'}{\partial \mathbf{h}_i^\top}\right\|_F = \|D_i W\|_F \le \|D_i\|_F \|W\|_F \le L_\sigma \|W\|_F. \tag{16}$$

Similarly, the second Frobenius norm can be simplified as:

$$\begin{aligned}\left\|\frac{\partial \mathbf{h}_i'}{\partial \mathbf{h}_j^\top}\right\|_F &= \left\|D_i W \cdot \frac{1}{k_i}\right\|_F = \frac{1}{k_i}\|D_i W\|_F \\ &\le \frac{\|D_i\|_F\|W\|_F}{k_i} \le \frac{L_\sigma \|W\|_F}{k_i}, \quad (\|D_i\|_F \le L_\sigma).\end{aligned}$$

Summing over the norms, we have:

$$\sum_{j\in\mathcal{N}(i)}\left\|\frac{\partial \mathbf{h}_i'}{\partial \mathbf{h}_j^\top}\right\|_F^2 \le k_i\left(\frac{L_\sigma\|W\|_F}{k_i}\right)^2 = \frac{(L_\sigma\|W\|_F)^2}{k_i}. \tag{17}$$

Step 4: Combine Terms for the Frobenius Norm
Combining Equation 16 and 17, we then have the Frobenius norm following the constraint:

$$\|\mathbf{J}\|_F^2 \le (L_\sigma\|W\|_F)^2\left(1+\frac{1}{k_i}\right). \tag{18}$$

Taking the square root:

$$\|\mathbf{J}\|_F \le L_\sigma\|W\|_F\sqrt{1+\frac{1}{k_i}}. \tag{19}$$

Step 5: Frobenius Norm Bound to Lipschitz Constant
Since the Lipschitz constant L is the supremum of the spectral norm of the Jacobian over all inputs $\|\mathbf{J}\|_2$, and the spectral norm is always bounded above by the Frobenius norm $\|\mathbf{J}\|_F$, any upper bound on the Frobenius norm immediately provides an upper bound on L:

$$L \le L_\sigma\|W\|_F\sqrt{1+\frac{1}{k_i}}. \tag{20}$$

□

C Proof of the Three Conditions for MCI

We define $p_b \in (0,1)$ as the baseline AUC and c as the absolute change in AUC. A valid (p_b, c) satisfies $p_b + c \in (0,1)$. We define MCI as:

$$\mathrm{MCI}(p_b, c) = \log\left(\frac{1-p_b}{1-p_b-c}\right). \tag{21}$$

We now prove that the design of MCI satisfies three conditions derived from three natural marginal utility comparison cases: (i) Under the same baseline performance (p_b), the larger the new performance is, the higher the cost is; (ii) Under the same ultimate performance ($p_b + c$), the lower the baseline performance is, the higher the cost is; (iii) Under the same absolute change in performance, the lower the baseline performance, the higher the cost required or lost (i.e., |MCI| is larger).

Proof. It is trivial to show that with the same p_b, the larger c is, the higher the value of MCI is, satisfying condition (i). The definition of MCI can be reformulated as:

$$\mathrm{MCI}(p_b, c) = \log\left(\frac{1-p_b}{1-(p_b+c)}\right).$$

From the above equation, it is trivial to tell that with the same $p_b + c$, the smaller p_b is, the larger the MCI is. Therefore, condition (ii) is satisfied.

We now prove that MCI satisfies condition (iii). Let $p_b^1, p_b^2 \in (0,1)$ be two baseline AUCs satisfying $p_b^1 < p_b^2$. Also consider a value for c such that $p_b^1 + c, p_b^2 + c \in (0,1)$. Therefore, (p_b^1, c) and (p_b^2, c) are valid pairs for MCI calculation. The difference between $\mathrm{MCI}(p_b^1, c)$ and $\mathrm{MCI}(p_b^2, c)$ can be formulated as:

$$\mathrm{MCI}(p_b^1, c) - \mathrm{MCI}(p_b^2, c) = \log\frac{(1-p_b^2)(1-p_b^1-c)}{(1-p_b^1)(1-p_b^2-c)}. \tag{22}$$

Subtracting the numerator from the denominator, we have:

$$(1-p_b^2)(1-p_b^1-c) - (1-p_b^1)(1-p_b^2-c) = c(p_b^2 - p_b^1).$$

Case 1: performance improvement with $c > 0$:
Since $c > 0$ and $p_b^2 > p_b^1$, we know $0 < c(p_b^2 - p_b^1) < 1$, which means the numerator is larger than the denominator in Equation 22. Therefore, we have:

$$\begin{aligned}&\log\left(\frac{(1-p_b^2)(1-p_b^1-c)}{(1-p_b^1)(1-p_b^2-c)}\right) > 0 \\ \Rightarrow\quad &\mathrm{MCI}(p_b^1, c) - \mathrm{MCI}(p_b^2, c) > 0 \\ \Rightarrow\quad &\mathrm{MCI}(p_b^1, c) > \mathrm{MCI}(p_b^2, c).\end{aligned}$$

Since we also have $p_b^{1,2} + c > p_b^{1,2}$, which means

$$\mathrm{MCI}(p_b^{1,2}, c) = \log\left(\frac{1-p_b^{1,2}}{1-(p_b^{1,2}+c)}\right) > 0.$$

Therefore, we have:

$$|\mathrm{MCI}(p_b^1, c)| > |\mathrm{MCI}(p_b^2, c)|,$$

indicating condition (iii) is satisfied in this case.

Case 2: performance decremethent with $c < 0$:
Since $c < 0$ and $p_b^2 > p_b^1$, we know $c(p_b^2 - p_b^1) < 0$, which means the numerator is smaller than the denominator in Equation 22. Therefore, we have:

$$\begin{aligned}&\log\frac{(1-p_b^2)(1-p_b^1-c)}{(1-p_b^1)(1-p_b^2-c)} < 0 \\ \Rightarrow\quad &\mathrm{MCI}(p_b^1, c) - \mathrm{MCI}(p_b^2, c) < 0 \\ \Rightarrow\quad &\mathrm{MCI}(p_b^1, c) < \mathrm{MCI}(p_b^2, c).\end{aligned}$$

Since we also have $p_b^{1,2} + c < p_b^{1,2}$, which means

$$\mathrm{MCI}(p_b^{1,2}, c) = \log\left(\frac{1-p_b^{1,2}}{1-(p_b^{1,2}+c)}\right) < 0.$$

Therefore, we have:

$$|\mathrm{MCI}(p_b^1, c)| > |\mathrm{MCI}(p_b^2, c)|,$$

indicating condition (iii) is satisfied in this case. □

References

[1] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive factorization network: Learning adaptive-order feature interactions. In *AAAI*.

[2] Yashar Deldjoo, Tommaso Di Noia, Eugenio Di Sciascio, and Felice Antonio Merra. 2020. How dataset characteristics affect the robustness of collaborative recommendation models. In *SIGIR*.

[3] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *WWW*.

[4] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

[7] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* (2015).

[8] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI*.

[9] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*.

[11] Roger A. Horn and Charles R. Johnson. 1985. *Matrix Analysis*. Cambridge University Press.

[12] Yelp Inc. 2022. Yelp Open Dataset. https://www.yelp.com/dataset.

[13] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *ACM Recommender Systems conference*.

[14] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[15] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *CIKM*.

[16] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *SIGKDD*.

[17] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving graph collaborative filtering with neighborhood-enriched contrastive learning. In *WWW*.

[18] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *CIKM*.

[19] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: an enhanced two-stream MLP model for CTR prediction. In *AAAI*.

[20] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. UltraGCN: ultra simplification of graph convolutional networks for recommendation. In *CIKM*.

[21] Carl Menger and Karl Menger. 1923. *Grundsätze der volkswirtschaftslehre*. Hölder-Pichler-Tempsky.

[22] Zhongyu Ouyang, Chunhui Zhang, Shifu Hou, Chuxu Zhang, and Yanfang Ye. 2024. How to Improve Representation Alignment and Uniformity in Graph-based Collaborative Filtering?. In *ICWSM*.

[23] Steffen Rendle. 2010. Factorization machines. In *ICDM*.

[24] J Ben Schafer, Joseph Konstan, and John Riedl. 1999. Recommender systems in e-commerce. In *ACM conference on Electronic commerce*.

[25] Valeriy Shevchenko, Nikita Belousov, Alexey Vasilev, Vladimir Zholobov, Artyom Sosedka, Natalia Semenova, Anna Volodkevich, Andrey Savchenko, and Alexey Zaytsev. 2024. From variability to stability: Advancing RecSys benchmarking practices. In *SIGKDD*.

[26] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *CIKM*.

[27] Zhen Tian, Ting Bai, Wayne Xin Zhao, Ji-Rong Wen, and Zhao Cao. 2023. EulerNet: Adaptive Feature Interaction Learning via Euler's Formula for CTR Prediction. In *SIGIR*.

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *ICLR*.

[29] Léon Walras. 1900. *Éléments d'économie politique pure: ou, Théorie de la richesse sociale*. F. Rouge.

[30] Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards Representation Alignment and Uniformity in Collaborative Filtering. In *SIGKDD*.

[31] Fangye Wang, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. Towards deeper, lighter and interpretable cross network for ctr prediction. In *CIKM*.

[32] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep and Cross Network for Ad Click Predictions. In *Proceedings of the Workshop on Data Mining for Online Advertising (ADKDD)*.

[33] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *WWW*.

[34] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*.

[35] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *SIGIR*.

[36] Shu Wu, Zekun Li, Yunyue Su, Zeyu Cui, Xiaoyu Zhang, and Liang Wang. 2025. GraphFM: Graph Factorization Machines for Feature Interaction Modelling. *Machine Intelligence Research* (2025).

[37] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *SIGIR*.

[38] Jieming Zhu, Qinglin Jia, Guohao Cai, Quanyu Dai, Jingjie Li, Zhenhua Dong, Ruiming Tang, and Rui Zhang. 2023. Final: Factorized interaction layer for ctr prediction. In *SIGIR*.