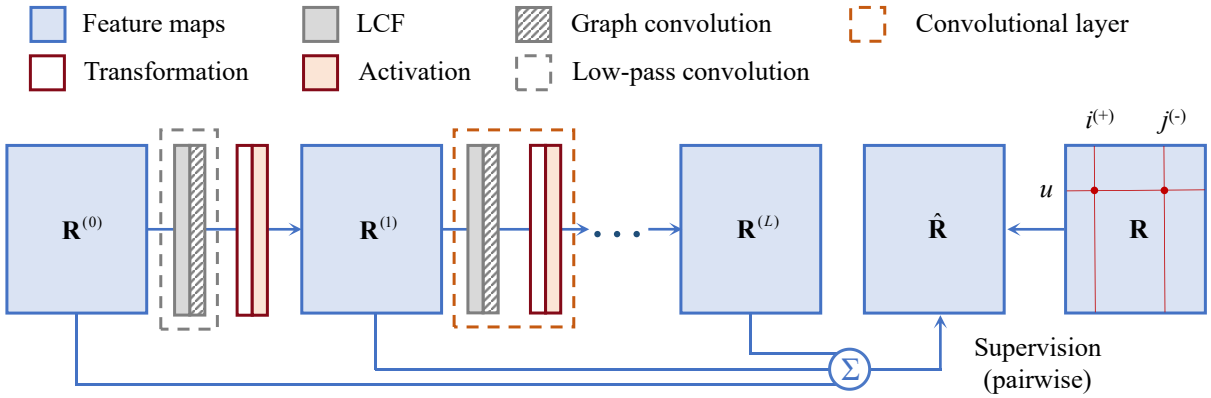

Figure 7. Illustration of LCFN model.

A. An Example to Understand LCF in the Recommendation Context

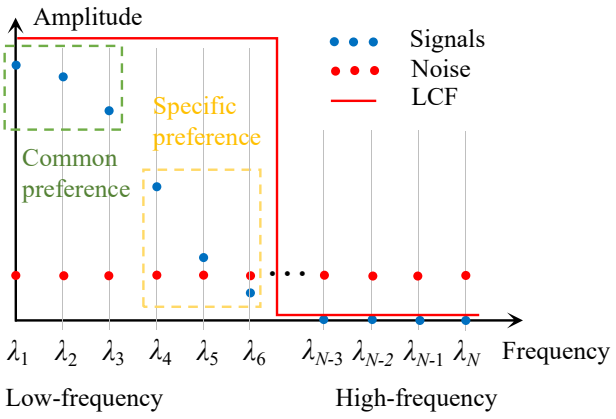

Figure 8. An intuitive examples for the mechanism of low-pass collaborative filter in recommendation.

Here, we give an example about Low-pass Collaborative Filter (LCF) to understand it intuitively in the recommendation context. For concise illustration, we introduce a 1D graph data on the user graph (i.e., user dimension) and 1D graph Fourier transform. For example, this data can be the i-th column in Figure 1(c), denoted as $\mathbf{R}_{*i}$. The graph Fourier transform $\mathcal{F}_g(\mathbf{R}_{*i})$ is shown in Figure 8. Blue points indicate the useful signal and red points indicate the noise. Since noise is randomly distributed pulses, it is white noise and (approximately) keeps constant in the frequency domain. According to the property of Laplacian matrix, $\lambda_1 = 0$ (Zhou et al., 2006), thus the first blue point is the direct current (DC) component of $\mathbf{R}_{*i}$. The DC component keeps constant among all users in the graph (time) domain hence indicates the popularity of item i intuitively, i.e., the global preference towards i. Signal in the green rectangle is the low-frequency component of $\mathbf{R}_{*i}$, which varies very smoothly among users and can be explained as the common preference of all users. The frequency of the signal in the yellow rectangle is higher, and this component varies more strongly in the graph domain, which can be explained as the specific (personalized) preference of users. Noise distributes on the high-frequency varies violently in the graph domain, which corrupt the signal thus should be removed by the low-pass collaborative filter. The low-pass filter is a gate function in the frequency domain (red line in Figure 8), which keeps the amplitude of low-frequency and reduces the amplitude of high-frequency to 0. We can see that by filtering, we remain the user preference and remove the noise.

B. Illustration of LCFN model

An illustration of LCFN is shown in Figure 7. $\mathbf{R}^{(0)}$ is the inputted matrix and $\{\mathbf{R}^{(l)}\}_{l=1\cdots L}$ are feature maps. We fuse them by sum pooling and supervise the output by the observed data.

C. Experiment Results

The performances of recommending top-{2, 5, 10, 20, 50, 100} items are shown in Table 4. We discuss several interesting observations. First, comparing *Amazon* and *Movielens*, it is obvious that the dataset with higher sparsity (*Amazon*) is more challenging to predict. However, LCFN gains more improvement on *Amazon* than on *Movielens*. The possible reason is that we face more serious sparsity issue on *Amazon* dataset, where the observed data cannot provide enough information to supervise model training thus we benefit a lot from exploring the topology.

Another observation also catches our interest is that the improvement LCFN gains reduces with the increasing of k. This may be because all models are tuned according to F_1-score@2, thus perform well for top-2 items recom-

Table 4. Recommendation performance (test set)

Datasets	Metrics (%)		MF	GCMC	NGCF	SCF	CGMC	LCFN	Improvement MF	Improvement BB
Amazon	F-1@	2	0.01356	0.01377	0.01349	0.01424	0.01363	**0.01654**	22.00%	16.18%
		5	0.01484	0.01452	0.01442	0.01459	0.01472	**0.01643**	10.69%	10.69%
		10	0.01355	0.01318	0.01352	0.01366	0.01350	**0.01465**	8.09%	7.20%
		20	0.01155	0.01114	0.01128	0.01126	0.01112	**0.01247**	7.92%	7.92%
		50	0.00833	0.00817	0.00810	0.00803	0.00805	**0.00902**	8.34%	8.34%
		100	0.00619	0.00607	0.00609	0.00599	0.00604	**0.00654**	5.70%	5.70%
	NDCG@	2	0.01742	0.01742	0.01733	0.01787	0.01728	**0.02104**	20.77%	17.71%
		5	0.02354	0.02362	0.02294	0.02343	0.02323	**0.02711**	15.15%	14.79%
		10	0.02940	0.02945	0.02943	0.02983	0.02944	**0.03373**	14.70%	13.05%
		20	0.03686	0.03657	0.03628	0.03673	0.03626	**0.04180**	13.42%	13.42%
		50	0.04828	0.04820	0.04724	0.04776	0.04719	**0.05445**	12.77%	12.77%
		100	0.05865	0.05850	0.05808	0.05806	0.05760	**0.06527**	11.30%	11.30%
Movielens	F-1@	2	0.07506	0.07689	0.07453	0.07808	0.07705	**0.08151**	8.60%	4.40%
		5	0.11531	0.11659	0.11533	0.11888	0.11789	**0.12133**	5.22%	2.06%
		10	0.14217	0.14372	0.14254	0.14513	0.14311	**0.14822**	4.26%	2.13%
		20	0.15728	0.15643	0.15815	0.16002	0.15727	**0.16250**	3.31%	1.55%
		50	0.15115	0.15028	0.15270	0.15262	0.14973	**0.15576**	3.05%	2.00%
		100	0.12843	0.12760	0.12976	0.12975	0.12671	**0.13151**	2.40%	1.34%
	NDCG@	2	0.25100	**0.26316**	0.25436	0.26173	0.25790	0.26129	4.10%	-0.71%
		5	0.23368	0.24114	0.23666	0.24191	0.24083	**0.24268**	3.85%	0.32%
		10	0.23292	0.23614	0.23473	0.23982	0.23724	**0.24285**	4.26%	1.26%
		20	0.25052	0.24955	0.25111	0.25595	0.25140	**0.26025**	3.88%	1.68%
		50	0.30066	0.29718	0.30041	0.30406	0.29815	**0.31237**	3.89%	2.73%
		100	0.35339	0.34841	0.35279	0.35600	0.34906	**0.36388**	2.97%	2.21%

mendation. However, they are not well tuned for a large k and advanced models fail to achieve the best improvement over the basic model. With the increasing of k, both LCFN and GNN/GCN baselines show less improvement over MF. To get better performance for a large k, we can retune all models according to F_1-score@100.