Initialization Matters: Regularizing Manifold-informed Initialization for Neural Recommendation Systems

Maria Wyrzykowska

Initialization Matters: Regularizing Manifold-informed Initialization for Neural Recommendation Systems

Yinan Zhang[1,3], Boyang Li[1,*], Yong Liu[2,3], Hao Wang[4], Chunyan Miao[1,2,3,*]
[1]School of Computer Science and Engineering, Nanyang Technological University
[2]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY)
[3]Alibaba-NTU Singapore Joint Research Institute [4]Alibaba Group
{yinan002,boyang.li,stephenliu,ascymiao}@ntu.edu.sg,cashenry@126.com

Motivation

Neural networks do not work

- SOTA neural recommendation approaches are difficult to reproduce and (often) perform worse than simple KNN methods.
- Initialization schemes seem to improve their potential:
  - Point of convergence is often close to the initialization.
  - Initialization can be a form of transfer learning.

→ The paper proposes initialization scheme based on Laplacian Eigenmaps, which represent "the intrinsic geometry of the data manifold".

Laplacian Eigenmaps

Notation

$\mathcal{D}$	Dimensionality of user and item embeddings
G	Graph used in Laplacian eigenmaps
$\mathcal{N}$	Number of nodes in the graph
W	Weighted adjacency matrix
$W_{i,j}$	Weight of the edge between nodes i and j
d_i	Degree of node i
d_{max}	Maximum degree of the graph
D	Diagonal degree matrix
L	Unnormalized graph Laplacian
λ_i	i-th eigenvalue of the graph Laplacian
q_i	i-th eigenvector of the graph Laplacian

KNN graph

To use Laplacian Eigenmaps, we first need to construct **KNN graphs** for users and items.
In the graph, we set $W_{i,j} = sim(i, j)$ if user j is among K nearest neighbours of user i or user i is among K nearest neighbours of user j.
Otherwise $W_{i,j} = 0$.
Similarity measure is **Jaccard index**:

$$sim(i, j) = \frac{|\mathcal{I}_i \cap \mathcal{I}_j|}{|\mathcal{I}_i \cup \mathcal{I}_j|} \tag{1}$$

Objective - intuition

We want to find set of vectors Q that minimize:

$$\sum_{q \in Q} \frac{1}{2} \sum_{i} \sum_{j} W_{i,j}(q_i - q_j)^2 \tag{2}$$

It means that if nodes i and j are close, then the values corresponding to them in each q are also close.
Additionally, the vectors q should be "different" from each other.

Graph Laplacian

Given an undirected graph $G =< V, E >$ with non-negative edge weights, Laplacian Eigenmaps creates $\mathcal{D}$-dimensional embeddings for the $\mathcal{N}$ graph nodes.
The **unnormalized graph Laplacian** is defines as $L = D - W$.
Alternatively we can use the **normalized symmetric graph Laplacian** $L_{sym} = D^{-1/2}(D - W)D^{-1/2}$.

Laplacian Eigenmaps

It can be shown that for all $q \in R^N$:

$$q^T L q = \frac{1}{2} \sum_i \sum_j W_{i,j} (q_i - q_j)^2 \tag{3}$$

and that eigenvectors $q^{(1)}, q^{(2)}, ..., q^{(\mathcal{D})}$ are solutions of the constrained minimization:

$$q^{(k)} = \underset{q}{\text{argmin}} \frac{1}{2} \sum_i \sum_j W_{i,j} (q_i - q_j)^2 \tag{4}$$

$$\text{s.t } q^T q = 1 \text{ and } q^T q^{(l)} = 0, \forall l < k$$

Nodes embeddings

Embeddings of the nodes of G are the rows of matrix $\hat{Q}$, which columns are the eigenvectors corresponding to $\mathcal{D}$ smallest eigenvalues of L.
The resulted embeddings represent multi-scale neighborhood information.

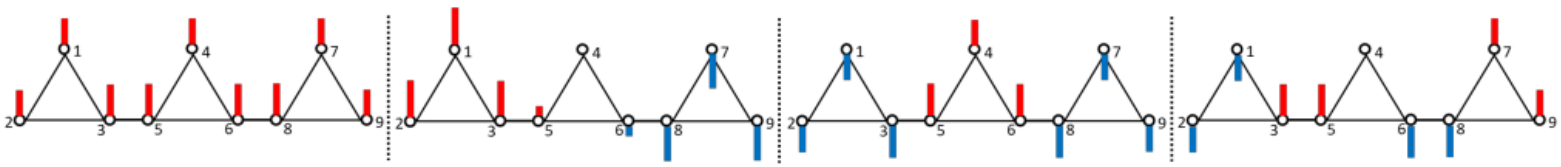
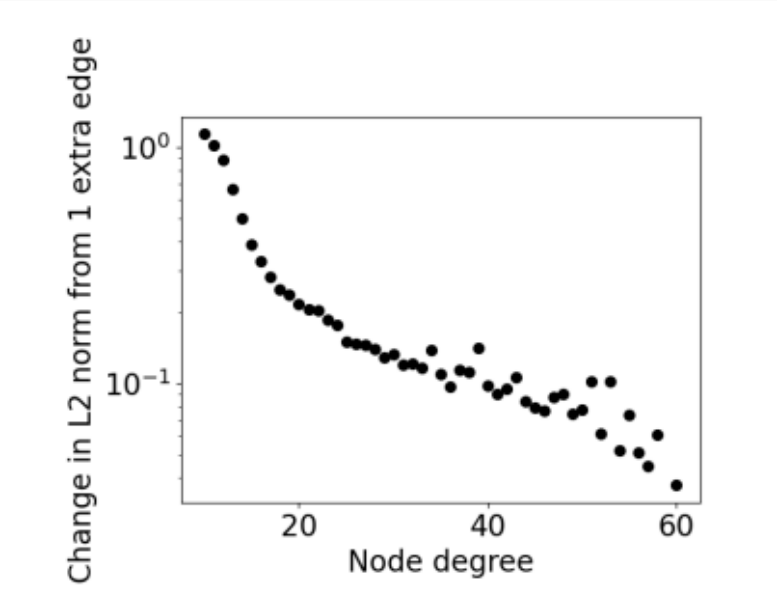


Figure 1: Node embeddings generated by Laplacian Eigenmaps (LE). We show the first four embedding dimensions sequentially (left to right) as bars overlaid on every node. Positive/negative values are denoted by red/blue, and the bar lengths indicate the absolute values. We observe greater fluctuation between neighboring nodes in later dimensions.

Popularity-based regularization

LE embeddings for relatively isolated nodes are unreliable - they can change drastically when new edges are added.



LEPORID selectively regularizes the embeddings of graph node i if d_i is small:

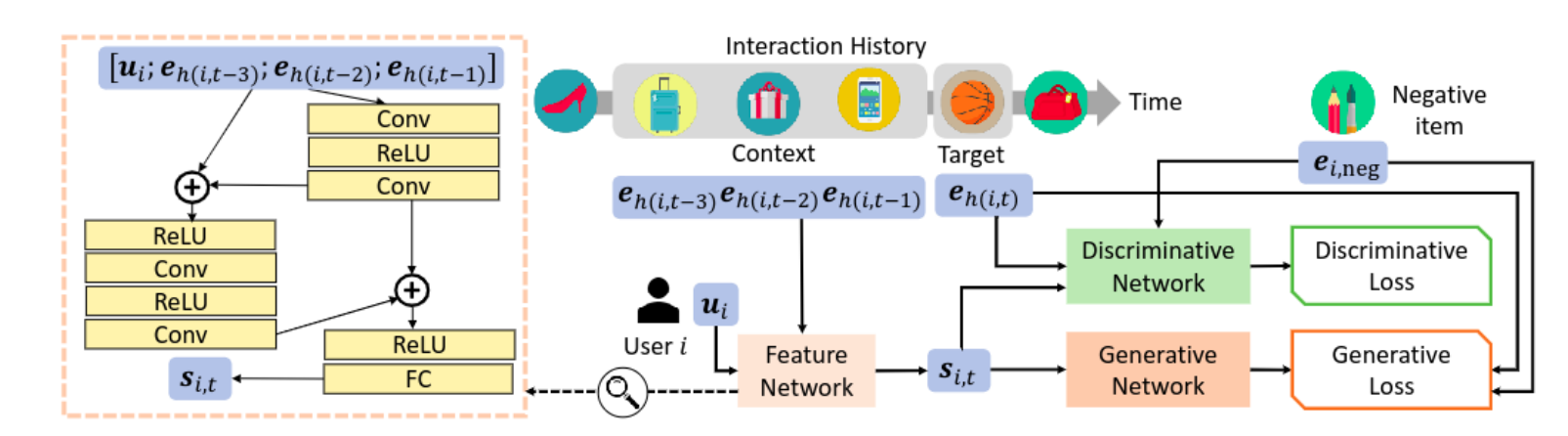
$$q^{(k)} = \underset{q}{\operatorname{argmin}} \frac{1}{2} \sum_i \sum_j W_{i,j}(q_i - q_j)^2 + \alpha \sum_i (d_{max} - d_i) q_i^2 \quad (5)$$

Computational complexity

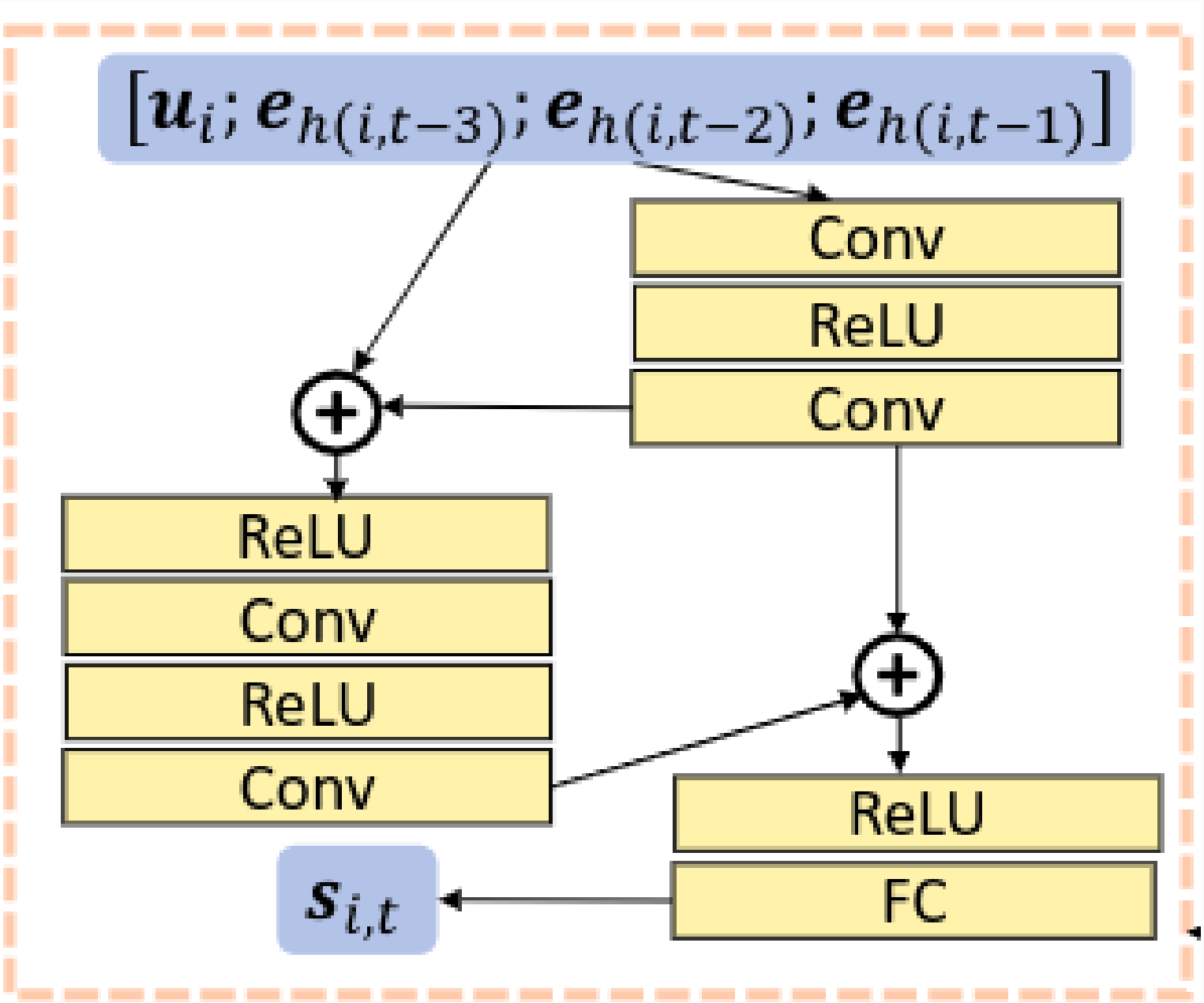
Eigenvalue decomposition has high time complexity of $O(\mathcal{N}^3)$, which does not scale well for large datasets. Fortunately, for KNN graph, Laplacian is a sparse matrix and there exist efficient algorithm utilizing this property: Implicit Restarted Lanczos Method (IRLM).
Calculating 64 eigenvectors for the biggest dataset that authors used took 172 seconds, whereas other simple model used for initialization (BPR) took 1888 seconds to converge.

Dual-Loss Residual Recommendation (DLR^2)

DLR^2 - architecture



Feature Network



Discriminative Network

The Discriminative Network $\mathcal{S}$ evaluates the preference of the user i towards item j at time t. Lower output value indicates higher similarity and better preference:

$$\mathcal{L}_{\mathcal{S}} = \mathbb{E}[\mathcal{S}(s_{i,t}, e_{h(i,t)})^2 + max(m_{\mathcal{S}} - \mathcal{S}(s_{i,t}, e_{i,neg}), 0))^2] \quad (6)$$

Generative Network

The Generative Network $\mathcal{G}$ directly predicts the embedding of items that the user prefers. The distance between the generated embedding $\mathcal{G}(s_{i,t})$ and the target item $e_{h(i,t)}$ should be smaller than the distance of the randomly sampled negative item $e_{i,neg}$:

$$\mathcal{L}_{\mathcal{G}} = \mathbb{E}[max(||\mathcal{G}(s_{i,t}) - e_{h(i,t)}||_2 - ||\mathcal{G}(s_{i,t}) - e_{i,neg}||_2 + m_{\mathcal{G}}, 0)] \quad (7)$$

Experiments

Datasets

Table 2: Dataset Statistics. Tail users and items are defined as the 25% of users and items with the least interaction records. Their data shares are the proportions of their interactions out of all interaction records.

Datasets	#Users	#Items	# Inter-actions	Data Density	Tail Users' Data Share	Tail Items' Data Share
ML-1M	6,040	3,650	1,000,127	4.54%	4.51%	1.08%
Steam	33,699	6,253	1,470,329	0.70%	11.67%	1.83%
Anime	47,143	6,535	6,143,751	1.99%	5.73%	0.93%

Baselines: initialization methods

- random
- SVD
- BPR
- node2vec
- NetMF
- Graph-Bert
- LE
- LEPORID

Baselines: neural recommenders

- NCF
- NGCF
- DGCF
- HGN
- DLR^2

Baselines: traditional recommenders

- TopPop
- ItemKNN
- UserKNN
- SLIM
- BPR

Results

- Different models and initialization methods evaluated using Hit Rate and F1 score
- LEPORID initialization consistently achieves the top position on most models, datasets and metrics
- UserKNN and ItemKNN are competitive to the neural approaches
- DLR^2 with LEPORID outperforms other methods, achieving especially high scores for tail users

Results

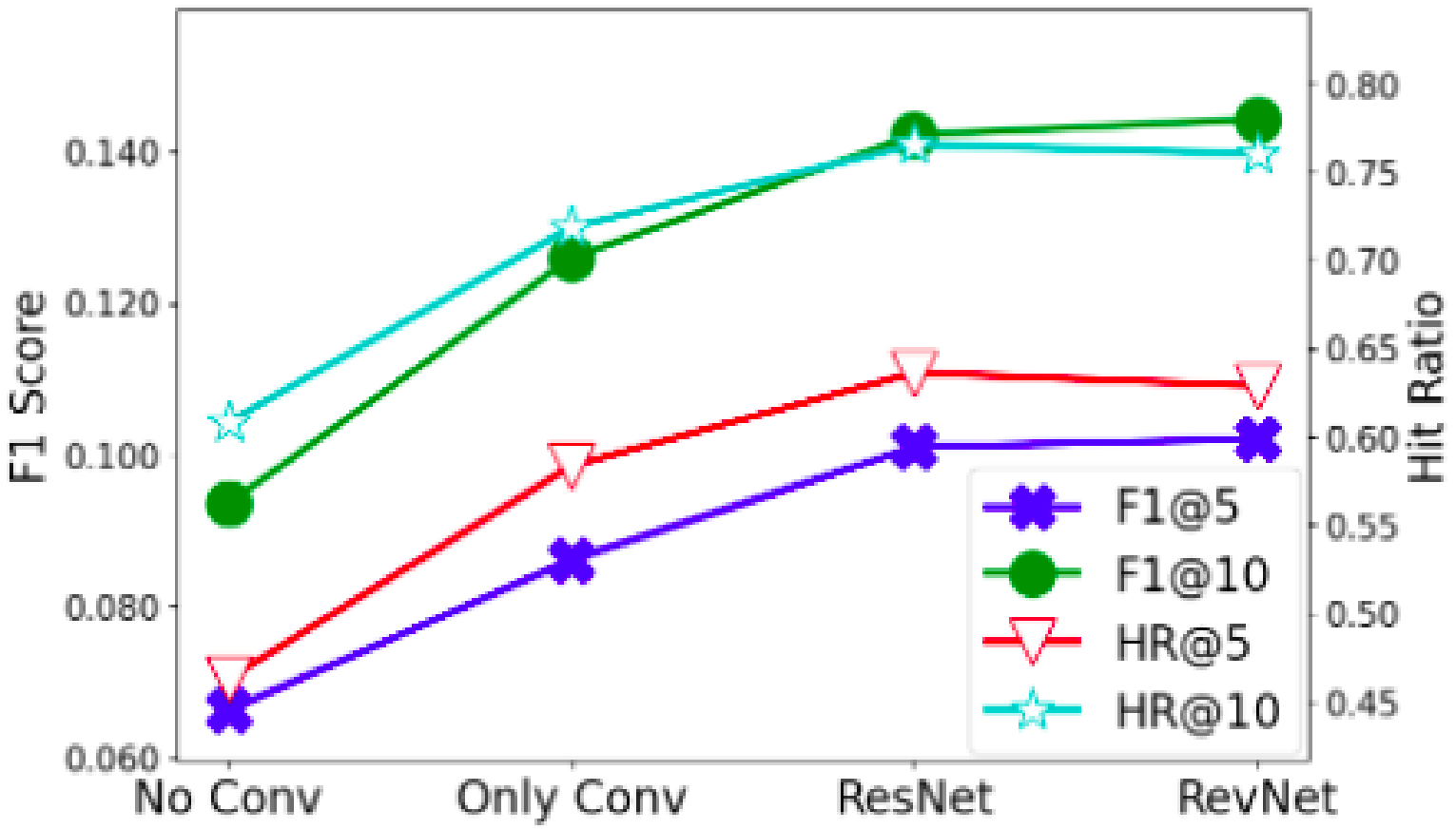
Methods	Initialization	Dataset: *ML-1M*				Dataset: *Steam*				Dataset: *Anime*			
		HR@5	HR@10	F1@5	F1@10	HR@5	HR@10	F1@5	F1@10	HR@5	HR@10	F1@5	F1@10
TopPop	–	0.3197	0.4377	0.0335	0.0517	0.1413	0.2306	0.0239	0.0307	0.4091	0.6103	0.0504	0.0807
UserKNN	–	0.3851	0.5419*	0.0483*	0.0760 *	0.1675	0.2575	0.0292	0.0357	0.4098	0.5899	0.0579	0.0802
ItemKNN	–	0.3868*	0.5235	0.0462	0.0682	0.1729*	0.2720*	0.0309*	0.0385*	0.5909*	0.7431*	0.0949*	0.1204*
SLIM	–	0.3692	0.5253	0.0435	0.0680	0.1671	0.2588	0.0290	0.0354	0.5337	0.7132	0.0684	0.1011
BPR	random	0.0851	0.1581	0.0107	0.0171	0.1582	0.2348	0.0279	0.0338	0.0406	0.0800	0.0074	0.0114
Relative Imp.		39.76%	29.65%	56.18%	52.38%	13.24%	10.07%	13.98%	13.39%	7.65%	3.00%	6.43%	18.04%
BPR	LEPORID	0.1094	0.2106	0.0151	0.0247	0.1638	0.2442	0.0293	0.0353	0.0810	0.1455	0.0170	0.0233
NCF	LEPORID	0.3262	0.4515	0.0311	0.0561	0.1027	0.2025	0.0167	0.0269	0.0678	0.1111	0.0054	0.0094
NGCF	LEPORID	0.3785	0.5258	0.0439	0.0667	0.1342	0.2084	0.0228	0.0277	0.5098	0.6401	0.0712*	0.1014*
DGCF	LEPORID	0.3588	0.5045	0.0425	0.0653	0.1661	0.2530	0.0292	0.0349	0.5264*	0.6477	0.0679	0.0916
HGN	LEPORID	0.3846 *	0.5475*	0.0543*	0.0797*	0.1822*	0.2745*	0.0318*	0.0380*	0.5010	0.6488*	0.0622	0.0902
Relative Imp.		40.56%	28.33%	38.98%	45.28%	7.46%	9.07%	10.93%	14.89%	20.84%	17.97%	41.84%	40.20%
DLR^2	LE	0.5119	0.6730	0.0692	0.1070	0.1853	0.2878	0.0332	0.0420	0.6065	0.7438	0.1010	0.1409
Relative Imp.		5.61%	4.40%	9.07%	8.26%	5.67%	4.03%	6.29%	3.85%	4.88%	2.90%	0.03%	0.86%
DLR^2	LEPORID	**0.5406**	**0.7026**	**0.0755**	**0.1158**	**0.1958**	**0.2994**	**0.0352**	**0.0437**	**0.6361**	**0.7654**	**0.1010**	**0.1421**

Ablation studies: loss function

Table 6: Performances of networks trained with different supervision on *ML-1M*.

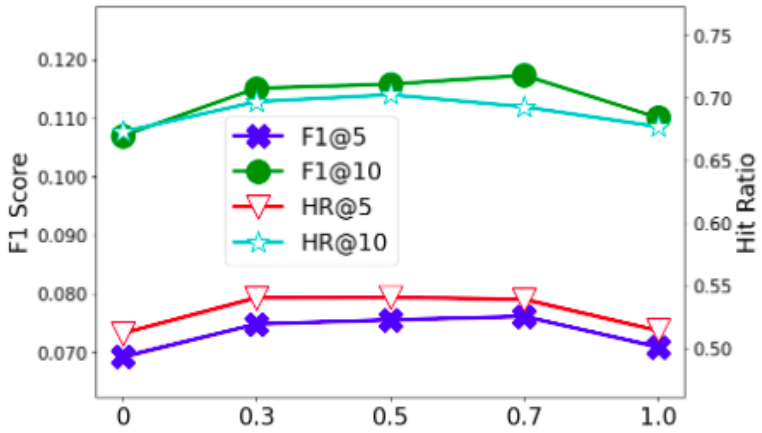
Inference Branch	Loss Function	HR@5	HR@10	F1@5	F1@10
$\mathcal{G}$	$\mathcal{L}_G$	0.3631	0.4982	0.0427	0.0601
	$\mathcal{L}_G + \mathcal{L}_S$	0.4955	0.6510	0.0677	0.1012
	Relative Imp.	36.46%	30.67%	58.66%	68.41%
$\mathcal{S}$	$\mathcal{L}_S$	0.5228	0.6826	0.0744	0.1112
	$\mathcal{L}_G + \mathcal{L}_S$	**0.5406**	**0.7026**	**0.0755**	**0.1158**
	Relative Imp.	3.40%	2.93%	1.49%	4.19%

Ablation studies: Feature Network architecture

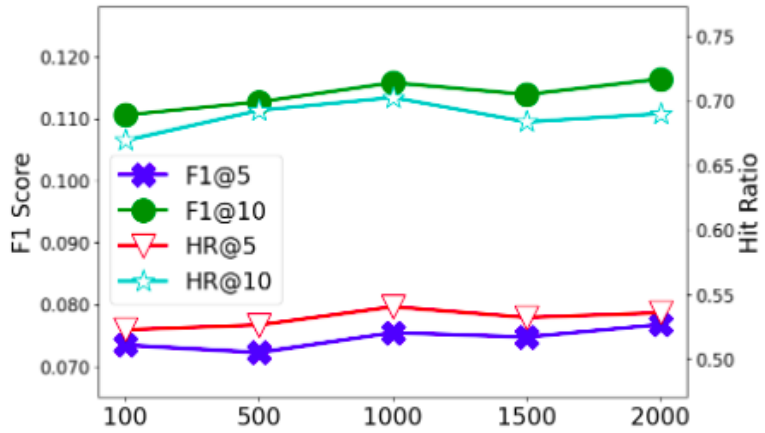


(a) Performance of different architectures in Feature Network in DLR^2 on *Anime* dataset, initialized by LEPORID.

Ablation studies: Hyperparameters



(b) Performance of different regularization coefficient α in LEPORID initialization on *ML-1M* dataset, evaluated on DLR^2.



(c) Performance of different neighbor K in LEPORID initialization on *ML-1M* dataset, evaluated on DLR^2.

Conclusion

Thank you for your attention!

Other links

On Laplacian Eigenmaps for Dimensionality Reduction - Juan Orduz

Laplacian Eigenmaps example