Learning Similarity Preserving Binary Codes for Recommender Systems

Yang Shi and Young-joo Chung

Rakuten Institute of Technology

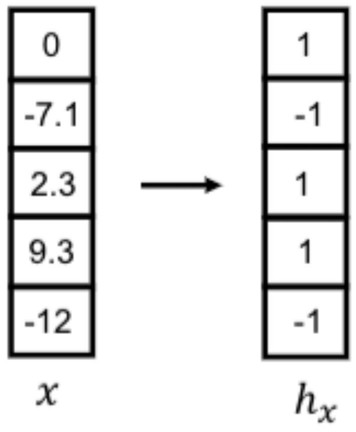
Rakuten, Inc.

8/15/2022

Rakuten

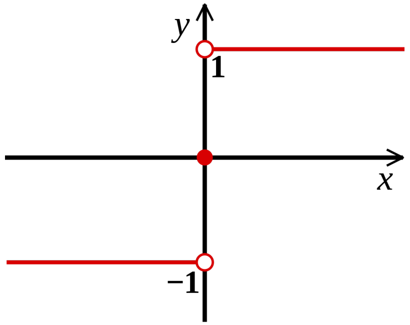
Motivation

Binary Hashing



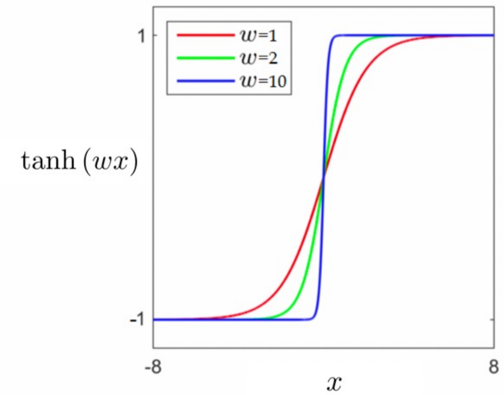
Binary hashing

- Hard threshold: sign function



Easy, can not backpropagate

- Soft threshold: scaled tanh function



Approximate, can backpropagate

Hashing-based recommender systems

- Reduce the memory requirement
- Accelerate the recommendation speed

Contributions

- Explore a new hashing-based RS design, Compact Cross-Similarity Recommender (CCSR), which is inspired by cross-modal retrieval literature.
- Demonstrate that Maximum a Posteriori (MAP)-based similarity loss works well in the top-k recommendation task.
- Analyze recommendation performance with different binarization methods. We show the simple $sign$ function still performs well compared to other more complicated methods.

Related work

Hashing-based recommender system:

- Feature extraction: Matrix Factorization, Auto Encoders, Neural Networks
- User-item interaction: Dot product(rating reconstruction), Cross Entropy (CE)
- Binarization: Sign, Linear Programming Relaxation (LPR), Straight-through-estimator (STE)

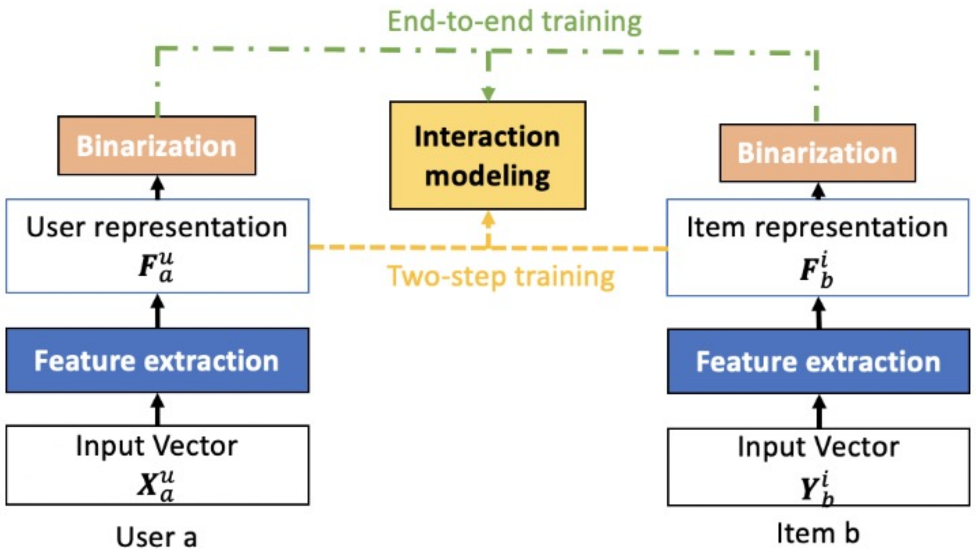


Figure 1: Hashing-based recommendation systems. Prediction is made by computing Hamming distances between binary codes.

Table 1: Comparison of different hashing-based recommender systems in terms of feature extraction, interaction modeling, and binarization

Paper	Loss function						Binarization
	Feature extraction			**User-item interaction**			
	MF	AE	Other NN	Dot product	CE	MAP	
CFCodeReg[27]	✓			✓			*Sign*
DCF[23]	✓			✓			*Sign*
NBR[25]	✓	✓		✓			LPR
NeuHash[8]	✓	✓		✓			STE
HashGNN[19]			✓		✓		*Sign*, STE
CCSR (ours)		✓				✓	*Sign*, Scaled *tanh*

CCSR

Feature Extraction

We use Auto encoders $\mathcal{L}_{ae} = \sum_{a,b} (||\mathbf{X}_a - \hat{\mathbf{X}}_a||_F^2 + ||\mathbf{Y}_b - \hat{\mathbf{Y}}_b||_F^2),$

User-item interaction and similarity

We use dot product between user and item to model user-item interaction/similarity

We use Maximum a Posteriori (MAP) estimation.

$$\log p(\mathbf{F}_a^u, \mathbf{F}_b^i | \mathbf{S}_{ab}) \propto \log p(\mathbf{S}_{ab} | \mathbf{F}_a^u, \mathbf{F}_b^i) p(\mathbf{F}_a^u) p(\mathbf{F}_b^i)$$

$$\mathcal{L}_{sim} = \sum_{a,b} (\log(1 + e^{\langle \mathbf{F}_a^u, \mathbf{F}_b^i \rangle}) - \mathbf{S}_{ab} \langle \mathbf{F}_a^u, \mathbf{F}_b^i \rangle).$$

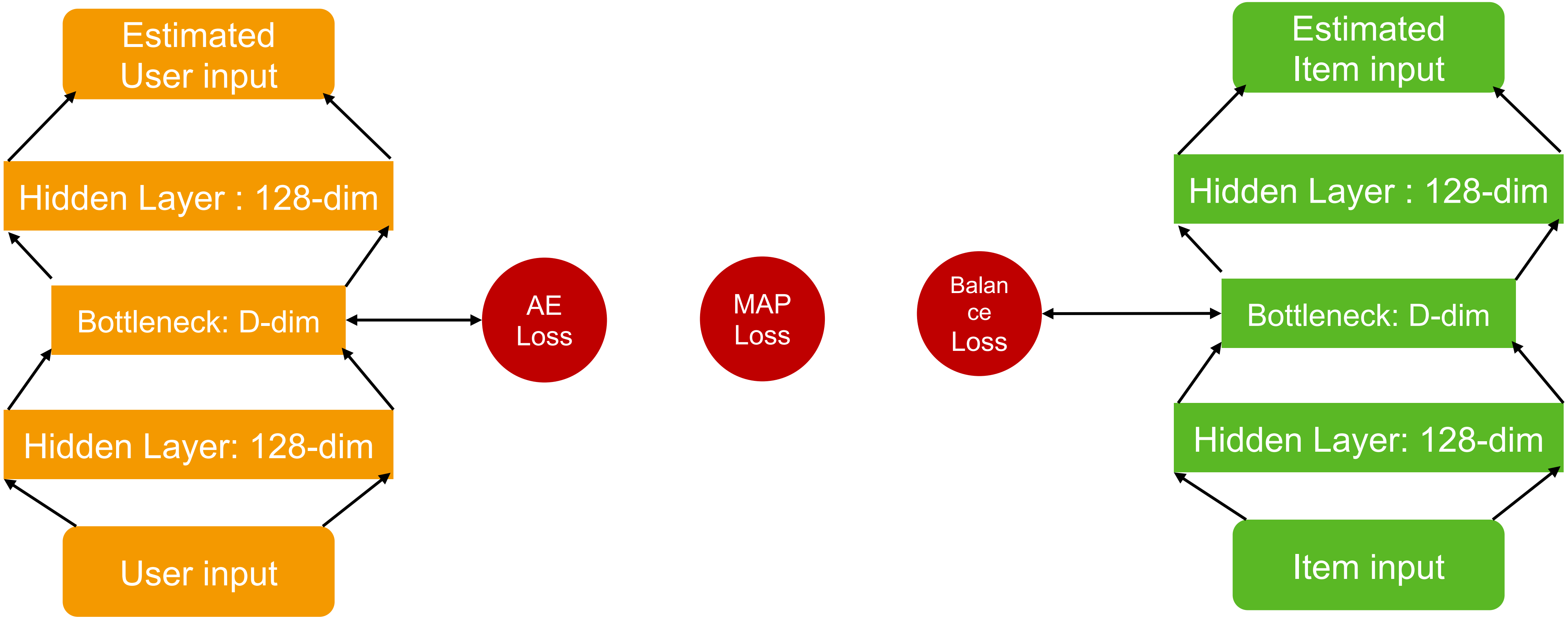
Additional loss: Balance loss: to balance the number of +1 and −1 in the binary code

$$\mathcal{L}_b = \sum_{a,b} (||\mathbf{F}_a^{u\top} \mathbf{1}||_F^2 + ||\mathbf{F}_b^{i\top} \mathbf{1}||_F^2)$$

Optimization: we minimize the all losses

$$\mathcal{L} = \mathcal{L}_{sim} + \lambda_b \mathcal{L}_b + \lambda_{ae} \mathcal{L}_{ae}$$

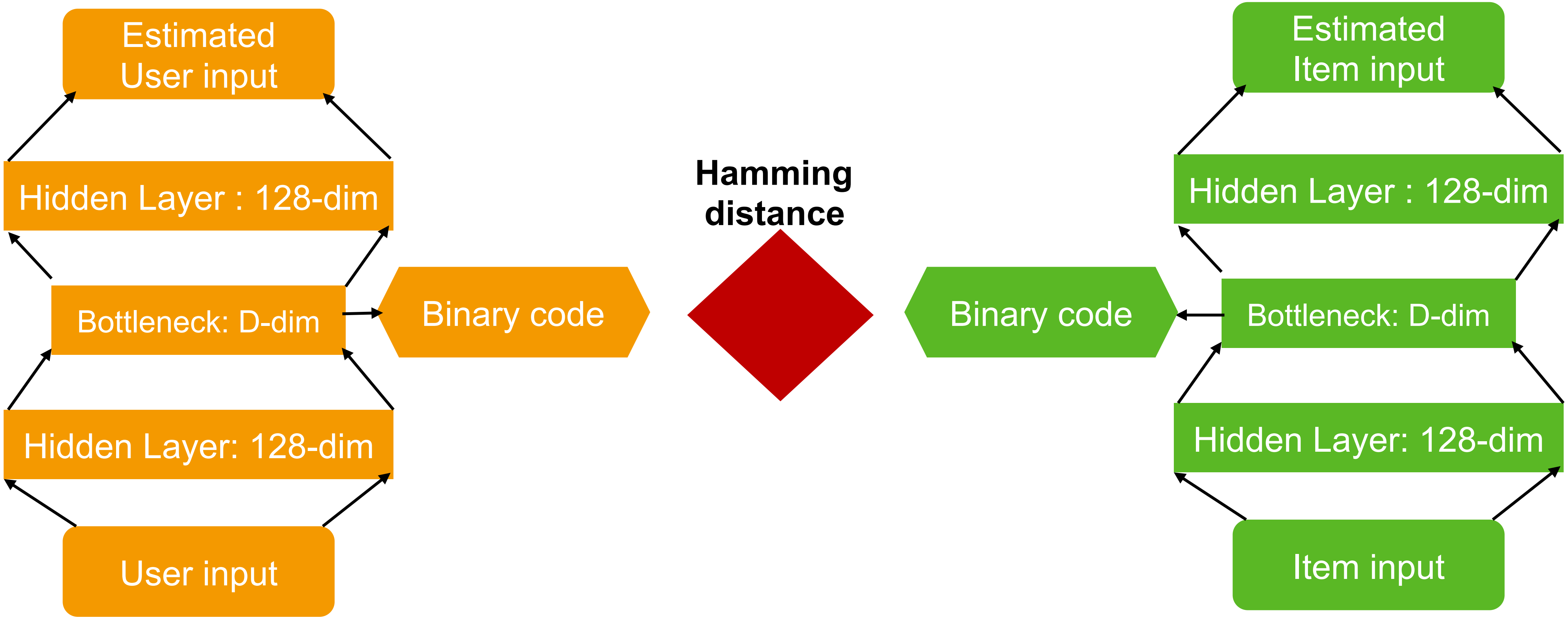
Model Training



e.g. target user’s ratings on all items

e.g. all user’s rating on target item

Model Inference



e.g. target user’s ratings on all items

e.g. all user’s rating on target item

Experiments

Datasets

Dataset	#User	#Item	#Ratings	Density
Movielens1M	6,040	3,952	1,000,209	4.19%
Amazon	35,736	38,121	1,960,674	0.14%
Ichiba	36,314	8,514	1,267,296	0.41%

Baselines

Rule-based
- Random
- Top

MF-based
- CF-S (Matrix Factorization CF)
- CFcodeReg
- AECF (AutoEncoder CF)

CR-based -- DJSRH

Binarization

- Sign $$h = sgn(f) = \begin{cases} 1 & f \geq 0 \\ -1 & f < 0, \end{cases}$$
- Scaled tanh $$h = tanh(\alpha f).$$
- Sign scaled tanh $$h = sgn(tanh(\alpha f)).$$

CCSR v.s. MF-based Hashing Recommenders

NDCG@k

	Models	@2				@6				@10			
		5	10	20	40	5	10	20	40	5	10	20	40
MovieLens	CF- S	0.5492	0.5599	0.5672	0.5833	0.6172	0.6198	0.6297	0.6450	0.6593	0.6613	0.6698	0.6840
	CFCodeReg	0.5692	0.5690	0.5738	0.5728	0.6314	0.6303	0.6323	0.6317	0.6712	0.6701	0.6724	0.6711
	AECF-S	0.4983	0.5555	0.5310	0.4874	0.5817	0.6154	0.5989	0.5689	0.6311	0.6534	0.6423	0.6175
	CCSR-S	**0.6332**	**0.6523**	**0.6912**	**0.7402**	**0.6997**	**0.7128**	**0.7277**	**0.7677**	**0.7371**	**0.7475**	**0.7529**	**0.7897**
Amazon	CF-S	0.7661	0.7673	0.7680	0.7691	0.8399	0.8409	0.8422	0.8428	0.8692	0.8700	0.8711	0.8716
	CFCodeReg	0.7657	0.7650	0.7653	0.7663	0.8398	0.8400	0.8405	0.8413	0.8692	0.8693	0.8698	0.8704
	AECF-S	0.7703	**0.7755**	**0.7818**	**0.7793**	0.8411	**0.8447**	**0.8481**	**0.8477**	0.8697	**0.8726**	**0.8756**	**0.8750**
	CCSR-S	**0.7707**	0.7685	0.7648	0.7752	**0.8420**	0.8408	0.8381	0.8446	**0.8705**	0.8697	0.8676	0.8725
Ichiba	CF-S	0.8915	0.8921	0.8927	0.8956	0.9329	0.9339	0.9338	0.9352	0.9475	0.9482	0.9483	0.9494
	CFCodeReg	0.8913	0.8911	0.8875	0.8856	0.9327	0.9322	0.9307	0.9300	0.9475	0.9470	0.9457	0.9452
	AECF-S	0.9150	0.9102	**0.9123**	0.9123	0.9454	0.9416	**0.9430**	0.9426	0.9566	0.9532	**0.9547**	0.9542
	CCSR-S	**0.9169**	**0.9104**	0.9026	**0.9158**	**0.9467**	**0.9432**	0.9384	**0.9478**	**0.9576**	**0.9548**	0.9513	**0.9583**

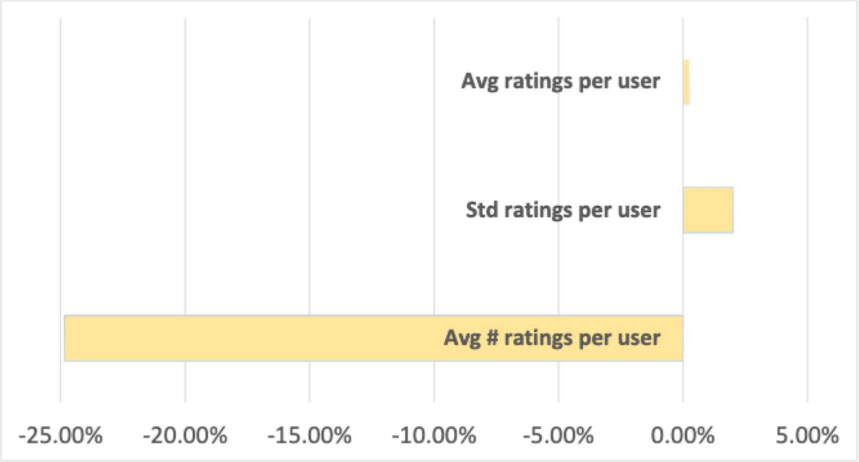
In MovieLens and Ichiba, CCSR performed best. In Amazon, CCSR performed second best.

=> CR-based CCSR worked better than MF-based Hashing recommenders in most cases

Similarity loss v.s. Rating reconstruction loss

To compare similarity (MAP) loss and rating reconstruction (MF) loss, we used continuous features for recommendation without binarization in MovieLens1M.

Two test user groups: one group obtained better results in NDCG@10 with CCSR-C, and the second group with the better results with AECF-C.



CCSR is helpful for users who rated less items and with higher rating variances.

Figure 3: Relative differences of the three characteristics between two user groups with code length 40 on Movielens1M.

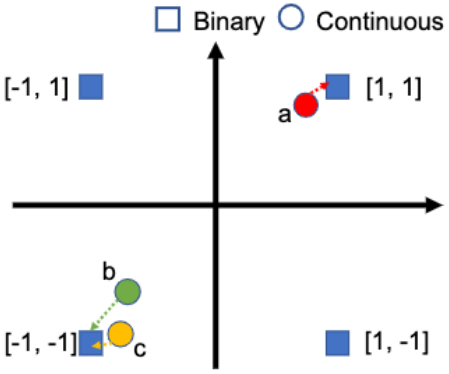
Reason: (1) AECF benefits from more ratings as it tries to **reconstruct original ratings** to learn the representations. (2) with higher rating variance, similarity-based models learn better representations **using similar and dissimilar pairs.**

Different Binarization Methods

NDCG@k of different models on Amazon

	Models	@2				@6				@10			
		5	10	20	40	5	10	20	40	5	10	20	40
Scaled $tanh$	AECF-ST	0.7800	0.7716	0.7810	0.7835	0.8471	0.8419	0.8478	0.8489	0.8748	0.8705	0.8752	0.8765
	DJSRH-ST	0.7460	0.7653	0.7707	0.7792	0.8263	0.8381	0.8412	0.8469	0.8579	0.8673	0.8701	0.8744
	CCSR-ST	**0.7833**	**0.7869**	**0.7837**	**0.7870**	**0.8490**	**0.8521**	**0.8503**	**0.8530**	**0.8762**	**0.8786**	**0.8770**	**0.8792**
Sign ST	AECF-SST	**0.7657**	0.7500	**0.7817**	**0.7733**	**0.8377**	0.8288	**0.8485**	**0.8432**	**0.8670**	0.8599	**0.8759**	**0.8714**
	DJSRH-SST	0.7610	0.7617	0.7627	0.7698	0.8356	**0.8372**	0.8379	0.8421	0.8654	**0.8668**	0.8676	0.8709
	CCSR-SST	0.7632	**0.7633**	0.7518	0.7530	0.8355	0.8362	0.8285	0.8306	0.8657	0.8658	0.8598	0.8614

Performance drops when we switch from Scaled tanh to Sign scaled tanh.



Limitation of SST

Here we have three continuous value features a, b, and c.
In ST models, continuous features are close to +1 and -1, and a is more similar to b than to c.
But b and c are equally similar to a after converted to binary codes using SST.

Take-away

- We found similarity-loss performs well on hashing-based top-k recommendation task
- Even though differentiable scaled $tanh$ is popular in recent discrete feature learning literature, a performance drop occurred when scaled $tanh$ outputs are forced to be binary.
- CCSR is helpful for users who rated less items and with higher rating variances

Thank you!