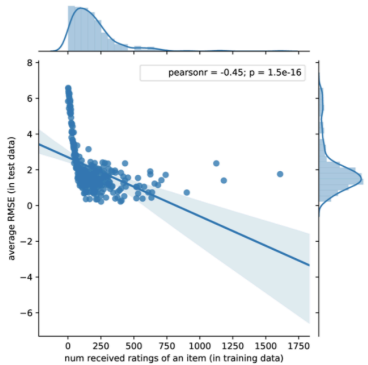
Differentiating regularization weights for recommender systems

Hung-Hsuan Chen

Cold start

- New items or the long tail items are rarely (or never) rated/viewed/purchased
- New users or less active users rarely rate/view/ purchase items
- We have limited information on these users and items
- Predictions are usually inaccurate, since we have limited (or no) information

Item's received rating counts in training data vs test RMSE



(a) The Epinions dataset

Pearson's correlation coefficient (PCC) on different datasets

Dataset	PCC	p-value
Epinions	-0.45	1.5×10^{-16} (***)
MovieLens-100K	-0.70	1.8×10^{-37} (***)
FilmTrust	-0.73	2.0×10^{-14} (***)
Yahoo! Movies	-0.12	0.04 (*)
AMI	-0.36	3.6×10^{-10} (***)

Key idea

- We have much information of the popular items and the active users
  - We probably should trust the data, since data size is large (less likely to overfit)
  - We should **assign lower constraints** when learning the vectors of these users and items
- We have little information of the long tail items and the less active users
  - We probably should make conservative predictions; trust the data too much may cause overfitting
  - We should **assign larger constraints** when learning the vectors of these users and items

Original loss function

- $$L(\Theta) = \frac{1}{2}\sum_{\forall(i,j)\in\tilde{k}}\left(r_{ij}-\hat{r}_{ij}\right)^2 + \frac{\lambda}{2}\|\Theta\|^2$$
  $$= \frac{1}{2}\sum_{\forall(i,j)\in\tilde{k}}\left(r_{ij}-\hat{r}_{ij}\right)^2 +$$
  $$\frac{\lambda}{2}\left\{\sum_{\forall i\in K_U}\|p_i\|^2 + \sum_{\forall i\in K_I}\|q_j\|^2 + \sum_{\forall i\in K_U}\|b_i\|^2 + \sum_{\forall i\in K_I}\|c_j\|^2\right\}$$
- $\hat{r}_{ij}$ can be predicted by Simon Funk's SVD, Factorization Machines, FPMC, etc.
- A universal regularization weight λ is assigned to all parameters

New loss function

- $$L(\Theta) = \frac{1}{2}\sum_{\forall(i,j)\in\tilde{k}}\left(r_{ij}-\hat{r}_{ij}\right)^2 + \frac{\lambda}{2}\left\{\sum_{\forall i\in K_U}\frac{1}{f(R^{(U)}(i))}\|p_i\|^2 + \sum_{\forall i\in K_I}\frac{1}{f(R^{(I)}(j))}\|q_j\|^2 + \sum_{\forall i\in K_U}\frac{1}{f(R^{(U)}(i))}\|b_i\|^2 + \sum_{\forall i\in K_I}\frac{1}{f(R^{(I)}(j))}\|c_j\|^2\right\}$$
  - $R^{(U)}(i)$: the number of items rated by user i
  - $R^{(I)}(j)$: the number of users who rated item j
  - $f(x)$ needs to be a positive and monotonically increasing function

Regularization terms

- $$\frac{\lambda}{2}\left\{\sum_{\forall i\in K_U}\frac{1}{f(R^{(U)}(i))}\|p_i\|^2 + \sum_{\forall i\in K_I}\frac{1}{f\left(R^{(I)}(j)\right)}\|q_j\|^2 + \sum_{\forall i\in K_U}\frac{1}{f(R^{(U)}(i))}\|b_i\|^2 + \sum_{\forall i\in K_I}\frac{1}{f\left(R^{(I)}(j)\right)}\|c_j\|^2\right\}$$
- When $R^{(U)}(i)$ is large, we have smaller constraints on p_i and b_i
- When $R^{(I)}(j)$ is large, we have smaller constraints on q_j and c_J

Proposed regularization functions

- Linear: $f(x) = ax + b\ (a > 0, b \geq 0)$
- Logarithm: $f(x) = \log(x + c)(\mathrm{c} > 0)$
- Square root: $f(x) = \sqrt{x + c}(\mathrm{c} \geq 0)$
- Support of functions: $x \geq 1$

Statistics of the benchmark datasets

Dataset	# users	# items	# ratings	Density	Rating scale
Epinions	40,163	139,738	664,824	0.0118%	[1, 2, 3, 4, 5]
MovieLens-100K	943	1,682	100,000	6.3047%	[1, 2, 3, 4, 5]
FilmTrust	1,508	2,071	35,497	1.1366%	[0.5, 1, 1.5, ..., 4]
Yahoo! Movies	7,642	11,916	221,367	0.2431%	[1, 2, ..., 13]
AMI	339,231	83,046	500,176	0.0018%	[1, 2, 3, 4, 5]

RMSE scores of SVD and SVD with regularization weights on the test datasets

Dataset	SVD	linear-reg	sqrt-reg	log-reg	improve ratio range
Epinions	1.1997	**1.0538**	**1.0538**	**1.0538**	12.16%
MovieLens-100K	0.9423	**0.9422**	**0.9422**	**0.9422**	0.01%
FilmTrust	0.8465	**0.8194**	**0.8194**	0.8223	2.86% to 3.20%
Yahoo! Movies	3.0799	**2.9892**	3.0129	3.0127	2.18% to 2.94%
AMI	1.1450	**1.1405**	**1.1405**	**1.1405**	0.39%

RMSE scores of SVD++ and SVD++ with regularization weights on the test datasets

Dataset	SVD++	linear-reg	sqrt-reg	log-reg	improve ratio range
Epinions	1.0628	**1.0538**	1.0566	**1.0538**	0.58% to 0.85%
MovieLens-100K	0.9423	**0.9422**	**0.9422**	0.9423	0.00% to 0.01%
FilmTrust	0.8199	0.8199	0.8197	**0.8194**	0.00% to 0.06%
Yahoo! Movies	3.0152	3.0127	**3.0126**	3.0128	0.08% to 0.09%
AMI	1.1446	**1.1407**	1.1429	1.1408	0.15% to 0.34%

RMSE scores of NMF and NMF with regularization weights on the test datasets

Dataset	NMF	linear-reg	sqrt-reg	log-reg	improve ratio range
Epinions	1.0709	**1.0582**	**1.0582**	1.0583	1.18% to 1.19%
MovieLens-100K	0.9621	0.9416	0.9416	**0.9413**	2.13% to 2.16%
FilmTrust	0.9295	**0.8206**	**0.8206**	**0.8206**	11.72%
Yahoo! Movies	3.6058	**3.0140**	3.0141	3.0141	16.41%
AMI	**1.1431**	1.1433	1.1432	1.1432	−0.01% to −0.02%

Various metrics of SVD and SVD + regularization weights on the Epinions dataset

Metrics	SVD	linear-reg	sqrt-reg	log-reg	improve ratio range
RMSE	1.1997	**1.0538**	**1.0538**	**1.0538**	12.16%
MAE	0.9139	**0.8190**	**0.8190**	**0.8190**	10.38%
R^2	0.0067	**0.2337**	**0.2337**	**0.2337**	3388.06%

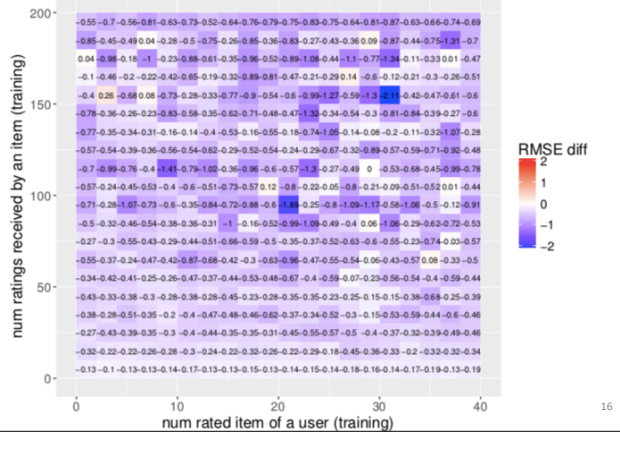
RMSE on Long Tail Items

Dataset	SVD	linear-reg	sqrt-reg	log-reg	improve ratio range
Epinions	2.2505	2.1189	**2.1186**	2.1187	5.85% to 5.86%
MovieLens-100K	1.8866	1.8781	**1.8785**	1.8808	0.31% to 0.45%
FilmTrust	1.8454	1.7246	**1.7204**	1.8122	1.80% to 6.77%
Yahoo! Movies	26.1175	**25.4348**	25.5993	25.5766	1.98% to 2.61%
AMI	3.0426	3.0436	**3.0406**	3.0448	−0.07% to 0.07%

Dataset	SVD++	linear-reg	sqrt-reg	log-reg	improve ratio range
Epinions	2.2295	2.1373	**2.1179**	2.1186	4.14% to 5.01%
MovieLens-100K	1.8809	1.8797	**1.8757**	1.8784	0.06% to 0.28%
FilmTrust	1.7523	1.7296	1.7337	**1.7180**	1.06% to 1.96%
Yahoo! Movies	26.2629	25.5760	**25.5558**	25.5704	2.62% to 2.69%
AMI	3.1112	**3.0448**	3.0480	3.0456	2.03% to 2.13%

Dataset	NMF	log-reg	linear-reg	sqrt-reg	improve ratio range
Epinions	2.0793	2.0757	2.0753	**2.0750**	0.17% to 0.21%
MovieLens-100K	2.0788	**1.8683**	1.8749	1.8724	9.81% to 10.13%
FilmTrust	1.8428	**1.6952**	1.7026	1.6956	7.61% to 8.01%
Yahoo! Movies	32.8463	**25.4302**	25.4408	25.4314	22.55% to 22.58%
AMI	2.9942	**2.9939**	2.9945	2.9950	−0.03% to 0.01%

RMSE diff b2n SVD & SVD+log RDF



Discussion

- Determine the **regularization weights** of the latent factors of the item and the user based on their **activeness** is helpful
  - Very simple, can be integrated with many learning-based recommendation models (e.g., SVD, SVD++, NMF, FM, FPMC, Prod2Vec, Behavior2Vec, etc.)
- Common wisdom: larger dataset ➔ smaller regularization weight
  - However, for recommender systems, a larger training dataset may **NOT** necessarily reveal more information of an item or an individual user (esp. long tail items and less active users)

If you are interested in details…

- Hung-Hsuan Chen and Pu Chen. "Differentiating Regularization Weights – a Simple Mechanism to Alleviate Cold Start in Recommender Systems." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 2019.
- GitHub code:
  - https://github.com/ncu-dart/rdf