SLIM: Sparse Linear Methods for Top-N Recommender Systems

Group 3: Kunj, Aditi, Jyoti, Garvika

based on the research paper by
Xia Ning and George Karypis

Top-N recommender systems

Memory-based Collaborative Filtering
Item-based k-Nearest-Neighbors: fast but low quality

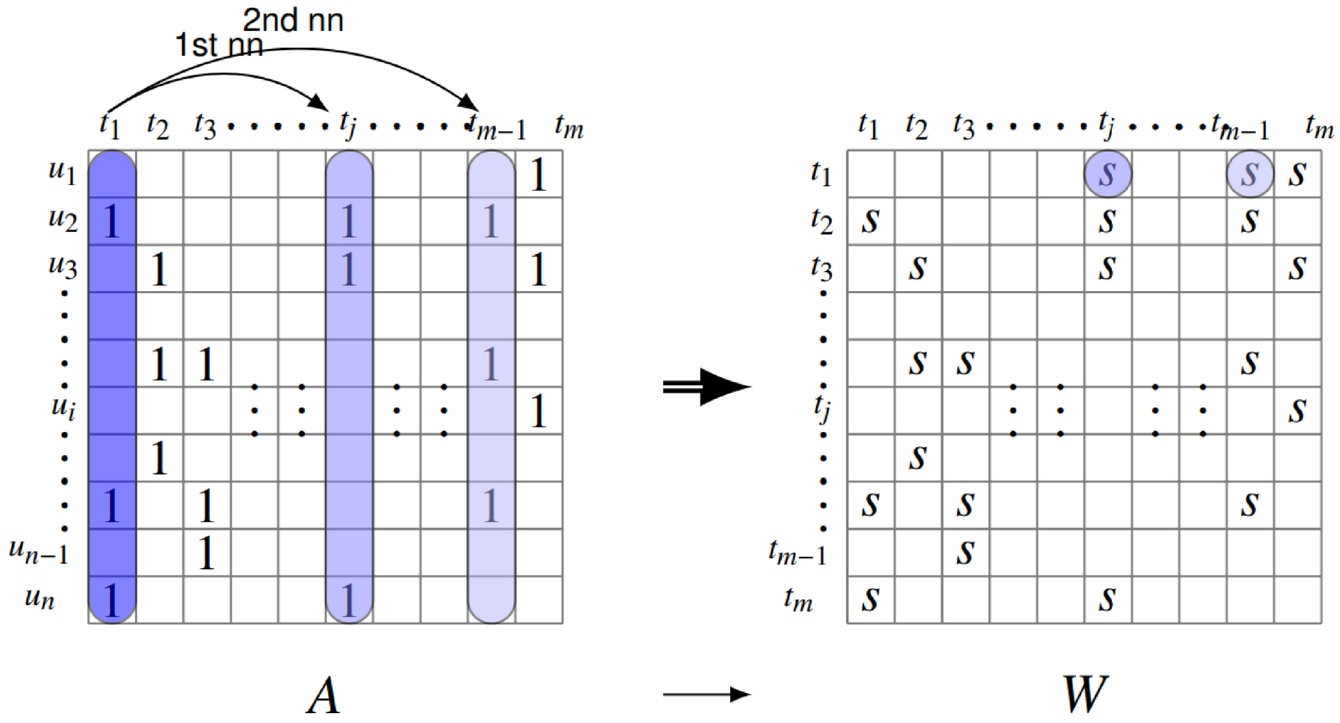
Model-based Collaborative Filtering
Singular Value Decomposition: high quality but slow

SLIM: Sparse LInear Methods
fast and high quality

Memory-based CF: itemkNN

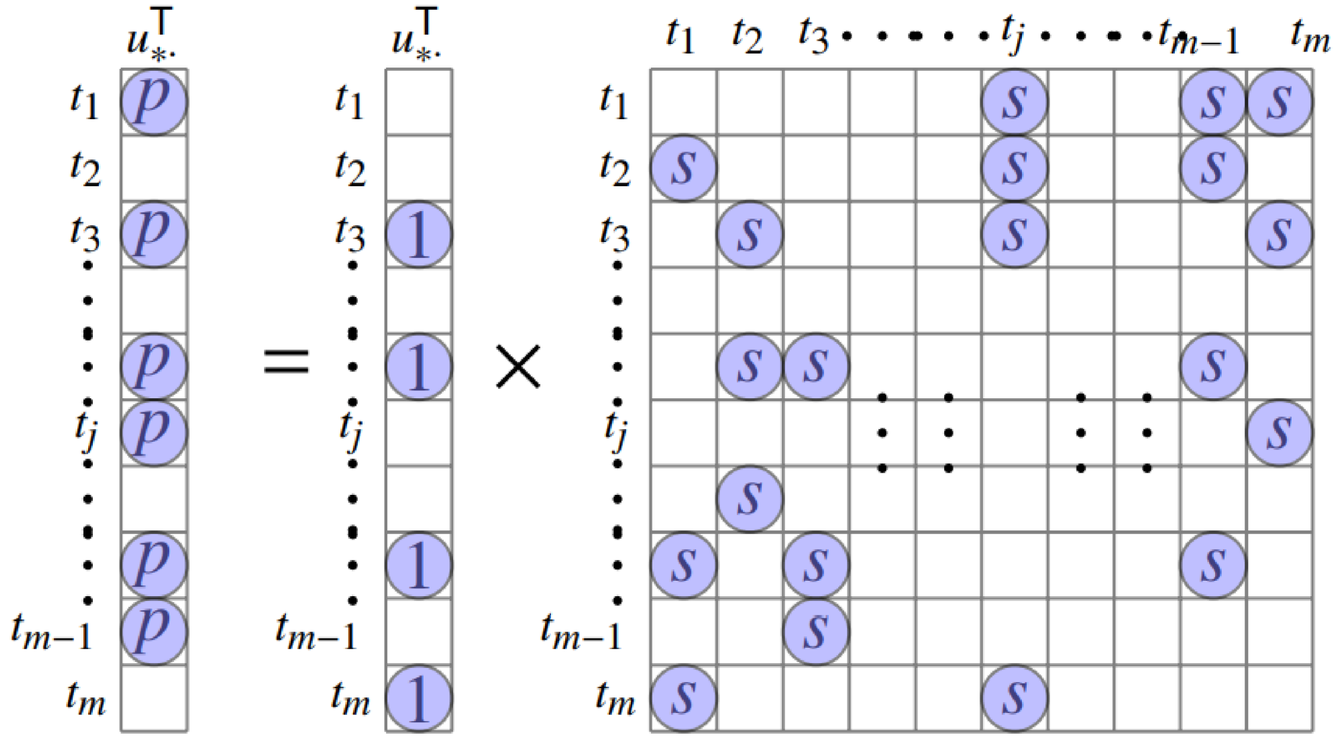
Identify a set of similar items

Item-item similarity calculated from A using cosine



Recommend similar items to what the user has purchased
Fast: sparse item neighborhood
Low quality: no knowledge is learned

$$\widetilde{a_i}^T = {a_i}^T + W$$



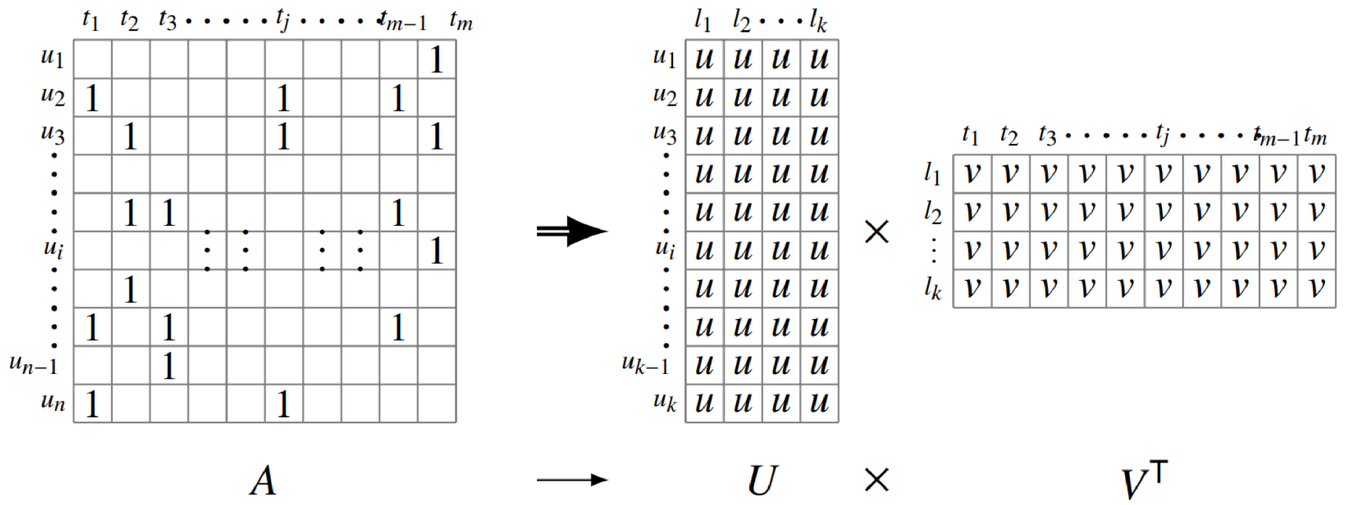
$$R(a, u_i) = \frac{\sum_j S(a, b) R(b, u_i)}{\sum_j S(a, b)}$$

$R(a, u_i)$: rating for movie **a** by user **u_i**
$S(a, b)$: similarity b/w movie **a** and **b**
$j \in J$, where **J** is the set of similar movies to **a**

Model-based CF: SVD

Factorize A into low-rank user and item factors that represent user and item characteristics in a common latent space
Formulated as an optimization problem

$$\underset{U,V^{\mathsf{T}}}{\text{minimize}} \quad \frac{1}{2}\|A - UV^{\mathsf{T}}\|_F^2 + \frac{\beta}{2}\|U\|_F^2 + \frac{\lambda}{2}\|V^{\mathsf{T}}\|_F^2$$

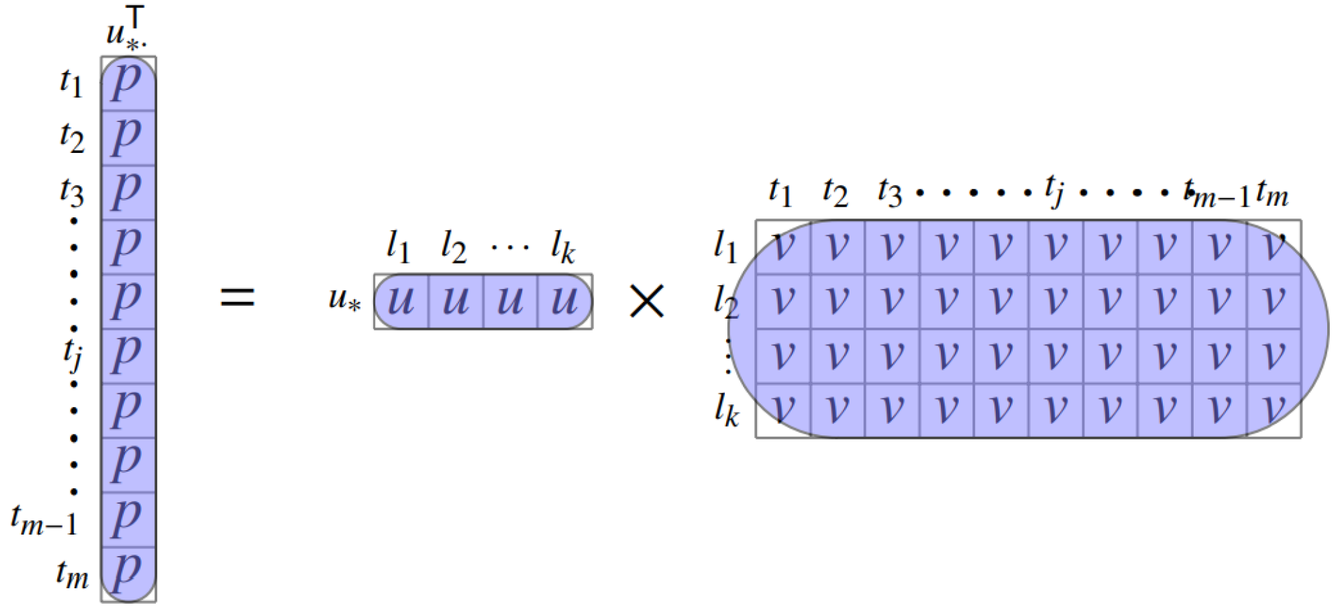


Prediction: dot product in the latent space

Slow: dense U and V^T

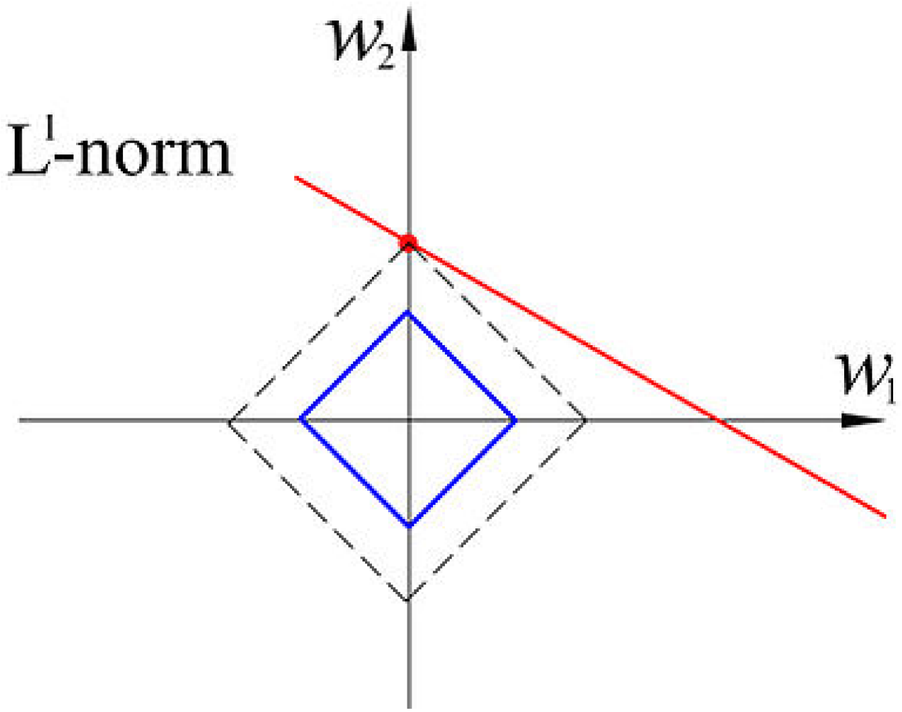
High quality: user tastes and item properties are learned

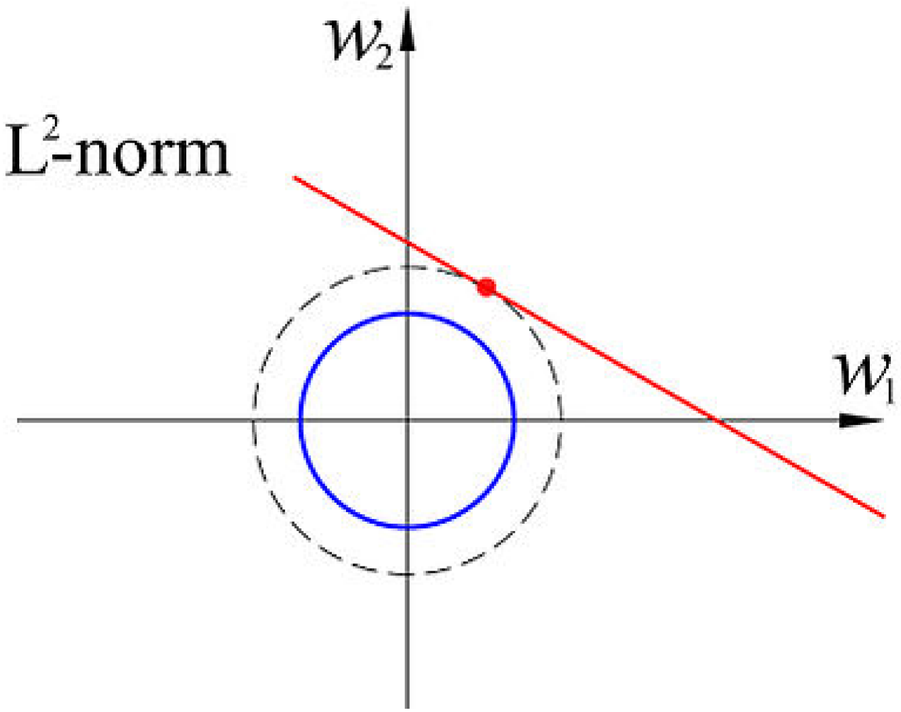
$$\widetilde{a_{ij}} = U_i^T . V_j$$



$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

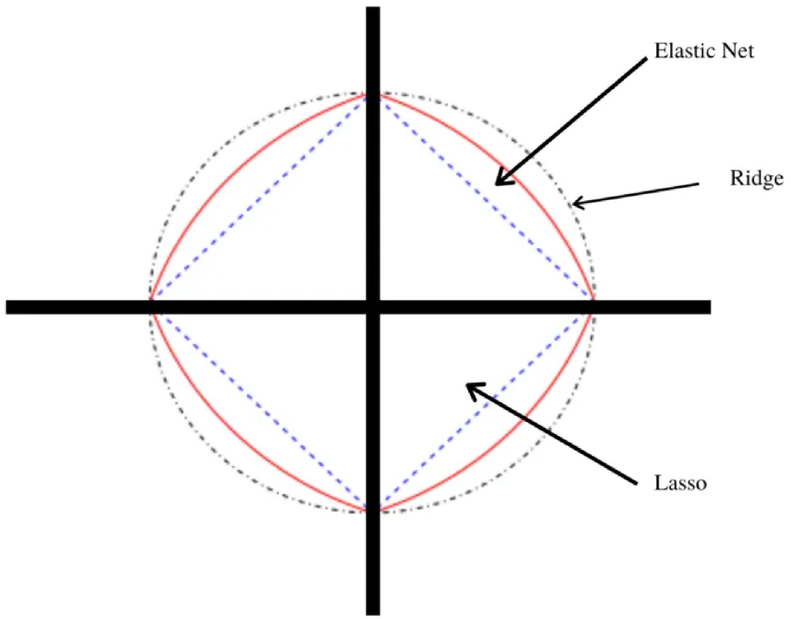




$$\hat{\beta}_{\text{lasso}} = \arg\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

$$\hat{\beta}_{\text{ridge}} = \arg\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 \right\}$$

$$\hat{\beta} = \arg\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda_2 \|\beta\|_2^2 + \lambda_1 \|\beta\|_1 \right\}$$



Minimize p_u and q_i over the loss function

Lasso

$$\min_{(p_u, q_i)} \sum_{(u,i) \in K} \left(r_{ui} - p_u^T q_i \right)^2 + \lambda_1 \left\| p_u \right\|_1 + \lambda_2 \left\| q_i \right\|_1$$

Ridge

$$\min_{(p_u, q_i)} \sum_{(u,i) \in K} \left(r_{ui} - p_u^T q_i \right)^2 + \rho_1 \left\| p_u \right\|_2^2 + \rho_2 \left\| q_i \right\|_2^2$$

Elastic Net

$$\min_{(p_u, q_i)} \sum_{(u,i) \in K} \left(r_{ui} - p_u^T q_i \right)^2 + \rho_1 \left\| p_u \right\|_2^2 + \rho_2 \left\| q_i \right\|_2^2 + \lambda_1 \left\| p_u \right\|_1 + \lambda_2 \left\| q_i \right\|_1$$

Motivations:
Recommendations generated fast
High quality recommendations

Key ideas:
retain the nature of itemkNN: sparse W
optimize the recommendation performance: learn W from A
sparsity structures
coefficient values

Def	Descriptions
u_i	user
t_j	item
$\mathcal{U}$	all users ($\|\mathcal{U}\| = n$)
$\mathcal{T}$	all items ($\|\mathcal{T}\| = m$)
A	user-item purchase/rating matrix, size $n \times m$
W	item-item similarity matrix/coefficient matrix
$\mathbf{a}_i^\mathsf{T}$	The i-th row of A, the purchase/rating history of u_i on $\mathcal{T}$
$\mathbf{a}_j$	The j-th column of A, the purchase/rating history of $\mathcal{U}$ on t_j

Row vectors are represented by having the transpose T supscript, otherwise by default they are column vectors.

$$
\begin{aligned}
\underset{W}{\text{minimize}} \quad & \frac{1}{2}\|A - AW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1 \\
\text{subject to} \quad & W \geq 0 \\
& \text{diag}(W) = 0,
\end{aligned}
$$

$$
\begin{aligned}
\underset{\mathbf{w}_j}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{a}_j - A\mathbf{w}_j\|_2^2 + \frac{\beta}{2}\|\mathbf{w}_j\|_2^2 + \lambda\|\mathbf{w}_j\|_1 \\
\text{subject to} \quad & \mathbf{w}_j \geq \mathbf{0} \\
& w_{j,j} = 0,
\end{aligned}
$$

Columns of W are independent, hence are easy to parallelize

Learning W using coordinate descent

- Objective function is separable & non-differentiable.
- Input features have sparse representations.
- If the dimensionality of the function is high, computing the gradient for all parameters can be very expensive.
- If the objective function is not smooth or has a discontinuity, then gradient descent may get stuck in the local minima or oscillate around the optimal solution. In such cases coordinate descent can be more robust as it sequentially updates.
- When input features are highly co-related, in such cases GD may suffer slow convergence due to the so-called zig-zagging effect

- Sparsity Introduction: The L1 regularization is known to introduce sparsity. This sparsity allows SLIM to generate recommendations very quickly.

- Model Complexity Control: The L2 regularization helps in controlling the complexity of the model and prevents overfitting. This leads SLIM to have better generalization performance.

- Implicit Grouping: The combination of L1 and L2 regularizations together implicitly groups correlated items in the solutions. This means that items that are related or have similar characteristics are grouped.

- Noise Reduction: The L1 regularization also helps in capturing the most informative signals while discarding noise due to sparsity. High-quality recommendations as only relevant information is considered.

Coordinate descent

This suggests that for the problem

$$\min_x \ f(x)$$

where $f(x) = g(x) + \sum_{i=1}^n h_i(x_i)$, with g convex and differentiable and each h_i convex, we can use coordinate descent: let $x^{(0)} \in \mathbb{R}^n$, and repeat

$$x_i^{(k)} = \underset{x_i}{\operatorname{argmin}} \ f\big(x_1^{(k)}, \ldots, x_{i-1}^{(k)}, x_i, x_{i+1}^{(k-1)}, \ldots, x_n^{(k-1)}\big),$$

$$i = 1, \ldots, n$$

for $k = 1, 2, 3, \ldots$

Important note: we always use most recent information possible

Consider the lasso problem:

$$\min_{\beta} \ \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$$

Note that the nonsmooth part is separable: $\|\beta\|_1 = \sum_{i=1}^{p} |\beta_i|$

Minimizing over β_i, with β_j, $j \neq i$ fixed:

$$0 = X_i^T X_i \beta_i + X_i^T (X_{-i}\beta_{-i} - y) + \lambda s_i$$

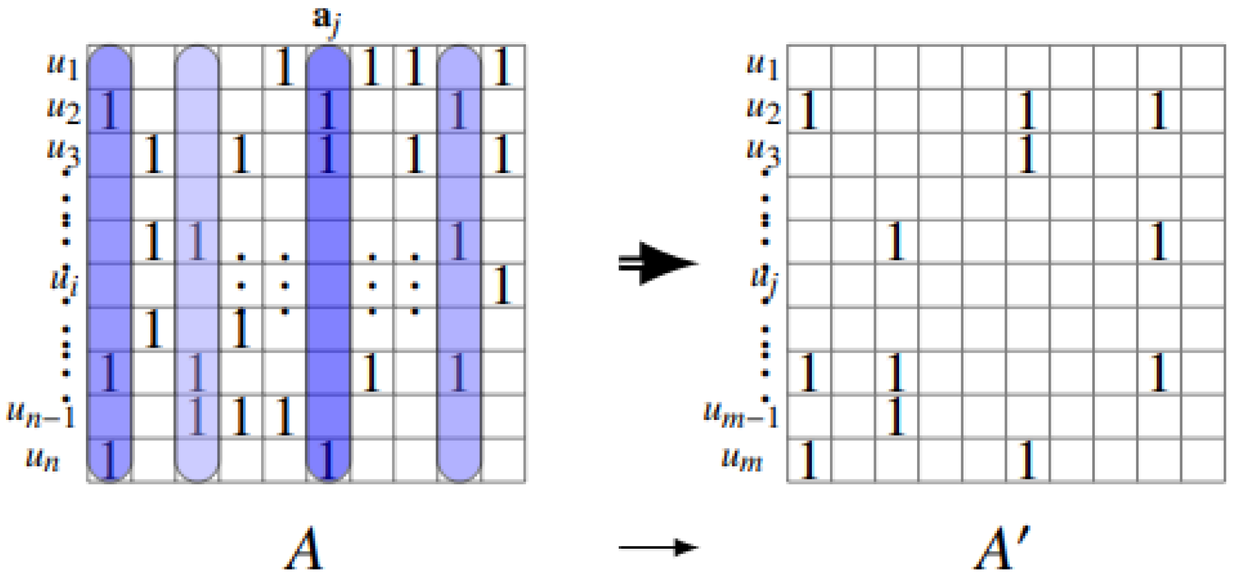
where $s_i \in \partial|\beta_i|$. Solution is simply given by soft-thresholding

$$\beta_i = S_{\lambda/\|X_i\|_2^2}\left(\frac{X_i^T(y - X_{-i}\beta_{-i})}{X_i^T X_i}\right)$$

Repeat this for $i = 1, 2, \ldots p, 1, 2, \ldots$

$$\underset{\mathbf{w}_j}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{a}_j - A\mathbf{w}_j\|_2^2 + \frac{\beta}{2}\|\mathbf{w}_j\|_2^2 + \lambda\|\mathbf{w}_j\|_1$$

- `fsSLIM`: `SLIM` with *f*eature *s*election
  - Prescribe the potential non-zero structure of $\mathbf{w}_j$
  - Select a subset of columns from A
    - `itemkNN` item-item similarity matrix



$$\underset{\mathbf{w}_j}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{a}_j - A'\mathbf{w}_j\|_2^2 + \frac{\beta}{2}\|\mathbf{w}_j\|_2^2 + \lambda\|\mathbf{w}_j\|_1$$

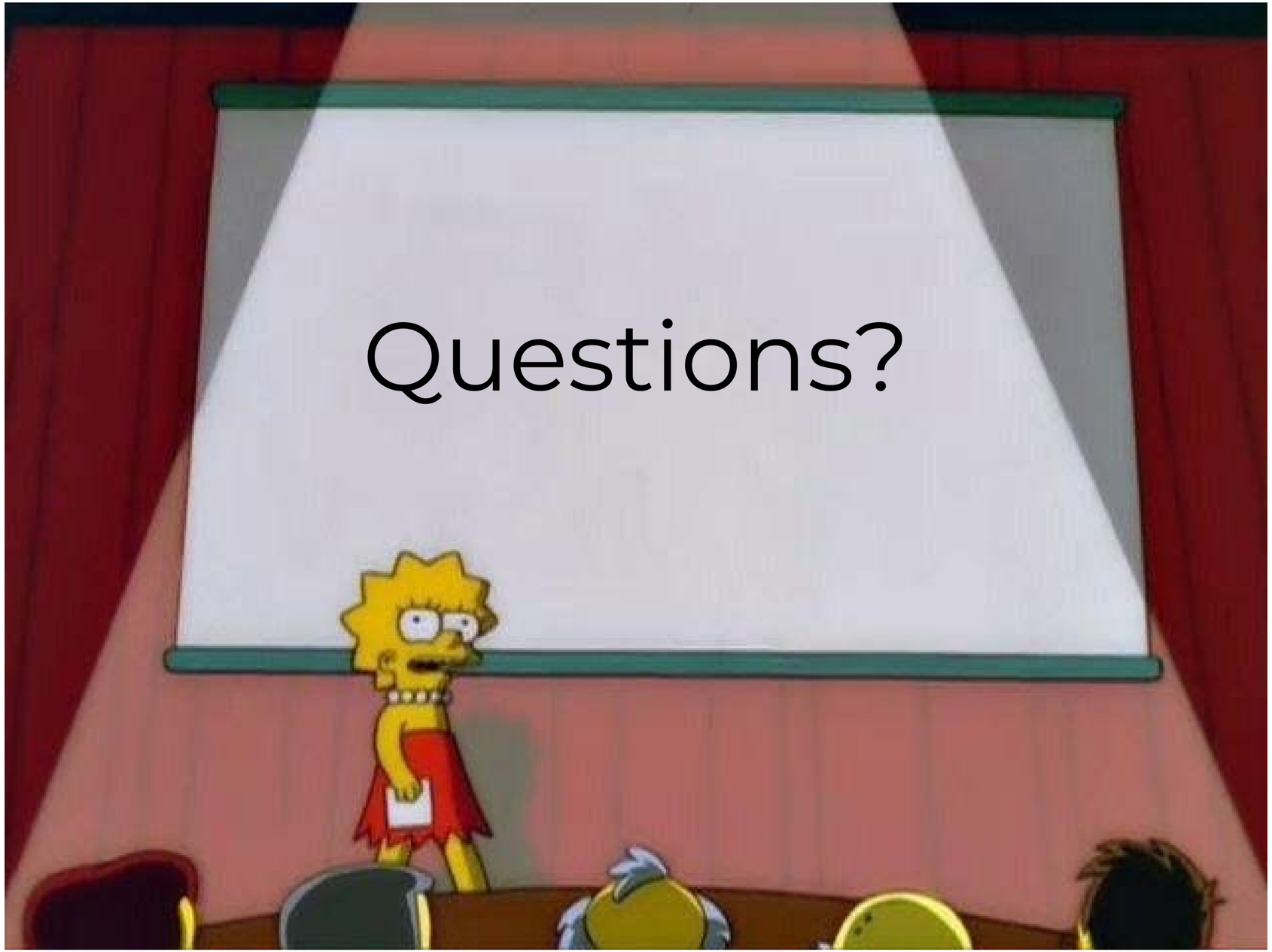
#users	#items	#ratings	rsize	csize	density
1292	313	23806	18.13	74.83	0.0589
2352	620	60976	25.55	96.92	0.0418
4203	1830	181442	42.65	97.97	0.0236
6046	3666	382038	62.52	103.11	0.0172
10213	12971	1276344	123.71	97.40	0.0096

Beer dataset with ~1.6 Million datapoints

Ratings from 1 to 5 with 0.5 granularity

- Evaluation methodology: Leave-One-Out cross validation
- Evaluation metrics
  - Hit Rate: $$\text{HR} = \frac{\#\text{hits}}{\#\text{users}}$$
  - Average Reciprocal Hit-Rank (ARHR) [2]:

$$\text{ARHR} = \frac{1}{\#\text{users}} \sum_{i=1}^{\#\text{hits}} \frac{1}{p_i}$$


Questions?