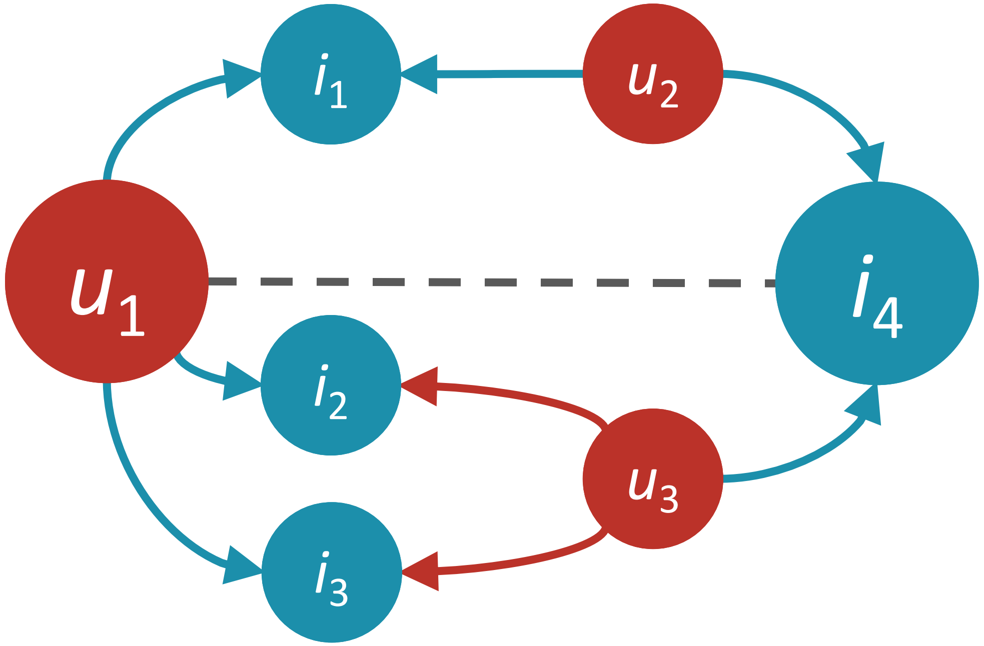
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Recommender Systems

Item-Based Collaborative Filtering

Rodrygo L. T. Santos
rodrygo@dcc.ufmg.br

User-based collaborative filtering



User-based collaborative filtering

	i_1	i_2	i_3	i_4
u_1	5	3	3	?
u_2	5			3
u_3		1	2	1

Breaking it down

Normalizing ratings

Computing similarities

Selecting neighborhoods

Aggregating ratings

How efficient is this?

User-based collaborative filtering

	i_1	i_2	i_3	i_4
u_1	1.3	-0.7	-0.7	?
u_2	1.0			-1.0
u_3		-0.3	0.7	-0.3

$\boldsymbol{m}$ users $\times$ $\boldsymbol{n}$ items

Cosine between two users: $O(n)$

Number of potential neighbors: $O(m)$

Cost per user: $O(mn)$

User-based collaborative filtering

	u_1	u_2	u_3
u_1	$s_{\vec{u}_1\vec{u}_1}$	$s_{\vec{u}_1\vec{u}_2}$	$s_{\vec{u}_1\vec{u}_3}$
u_2	$s_{\vec{u}_2\vec{u}_1}$	$s_{\vec{u}_2\vec{u}_2}$	$s_{\vec{u}_2\vec{u}_3}$
u_3	$s_{\vec{u}_3\vec{u}_1}$	$s_{\vec{u}_3\vec{u}_2}$	$s_{\vec{u}_3\vec{u}_3}$

$\boldsymbol{m}$ users $\times$ $\boldsymbol{n}$ items

Cosine between two users: $O(n)$

Number of potential neighbors: $O(m)$

Cost per user: $O(mn)$

Cost for all users: $O(m^2n)$

Problem: m and n in the order of 10^7

Some simple optimizations

Can exploit matrix symmetry

◦ Only need to compute one direction

Can exploit matrix sparsity

◦ Only need to consider users with a common item

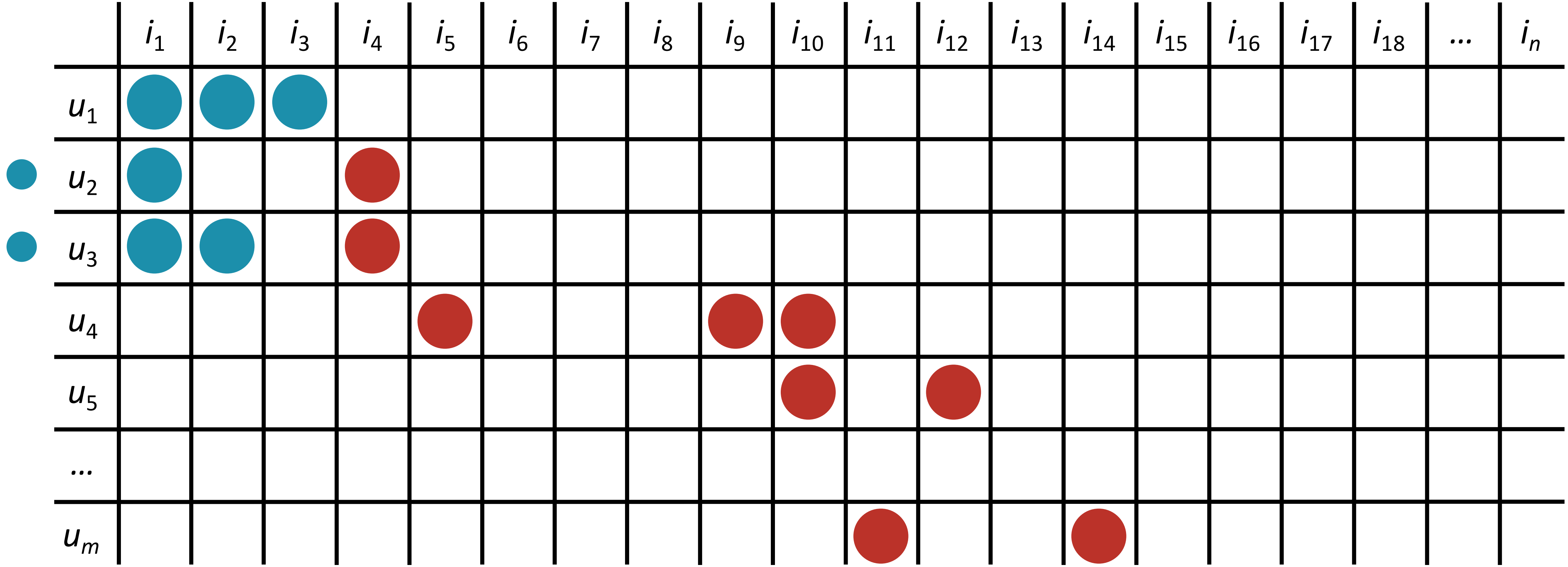
Still a costly operation to perform online

◦ ***Should we precompute similarities?***

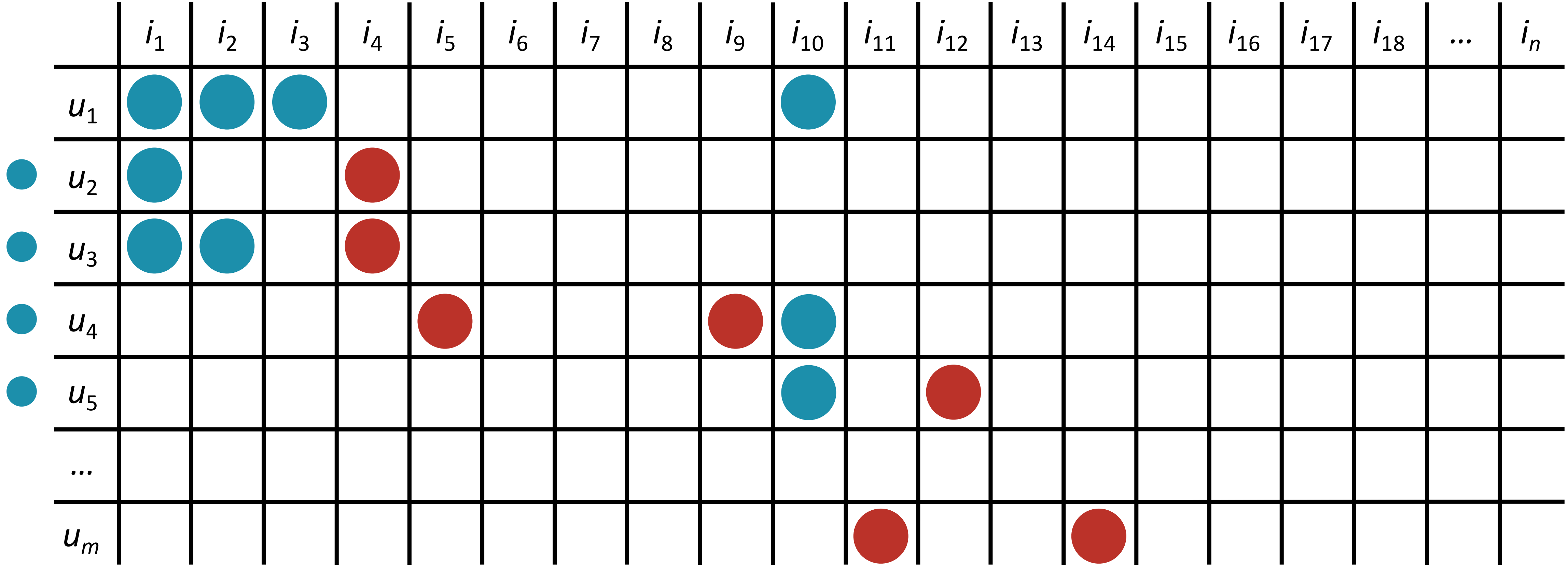
User-based collaborative filtering

	i_1	i_2	i_3	i_4
u_1	1.3	-0.7	-0.7	?
u_2	1.0			-1.0
u_3		-0.3	0.7	-0.3

User-based collaborative filtering



User-based collaborative filtering



User representation stability

User vectors are **extremely sparse**

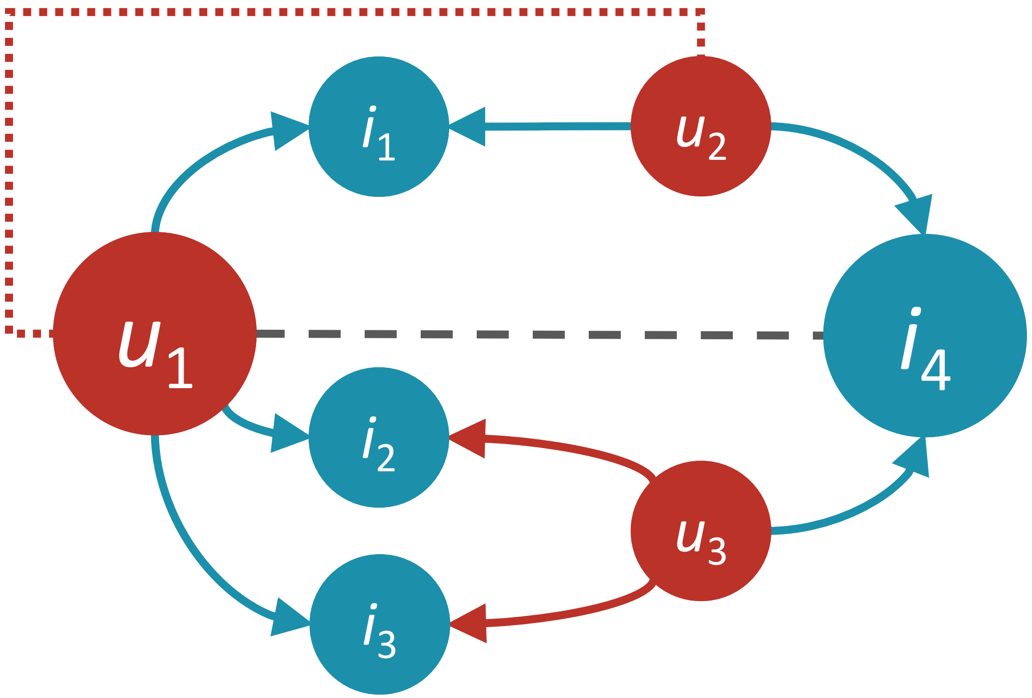
- Typical user consumes a tiny fraction of all items

User vectors are **unstable**

- A new feedback may drastically reposition the user in the vector space with respect to other users

Precomputed user similarities quickly become stale

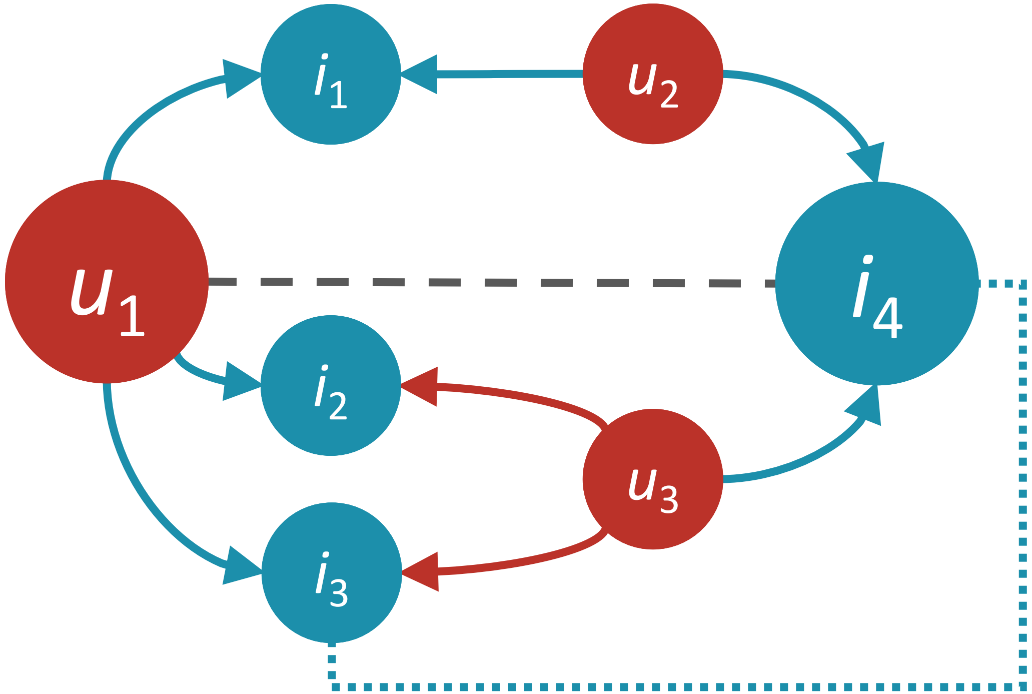
User-based collaborative filtering



similar users tend to like the same items

recommend items liked by users similar to u_1

Item-based collaborative filtering

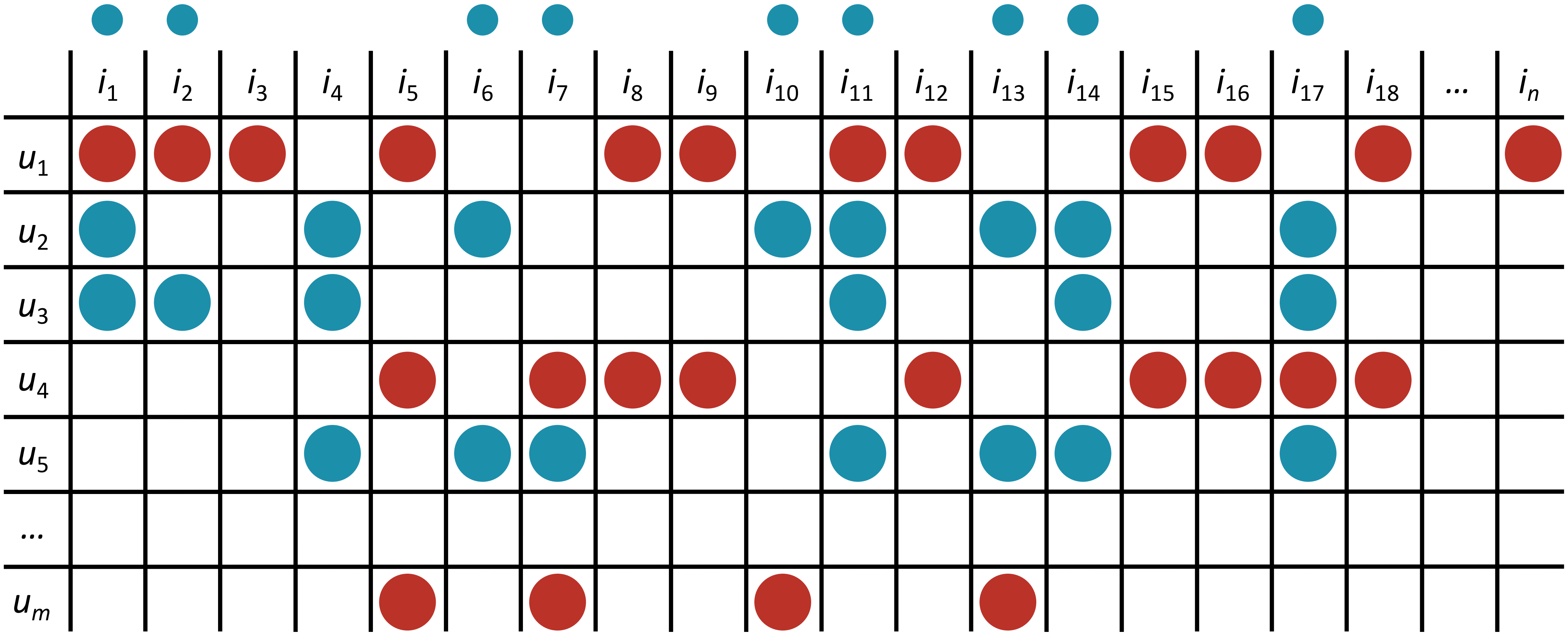


users who like an item
tend to like similar items
recommend items similar
to those consumed by u_1

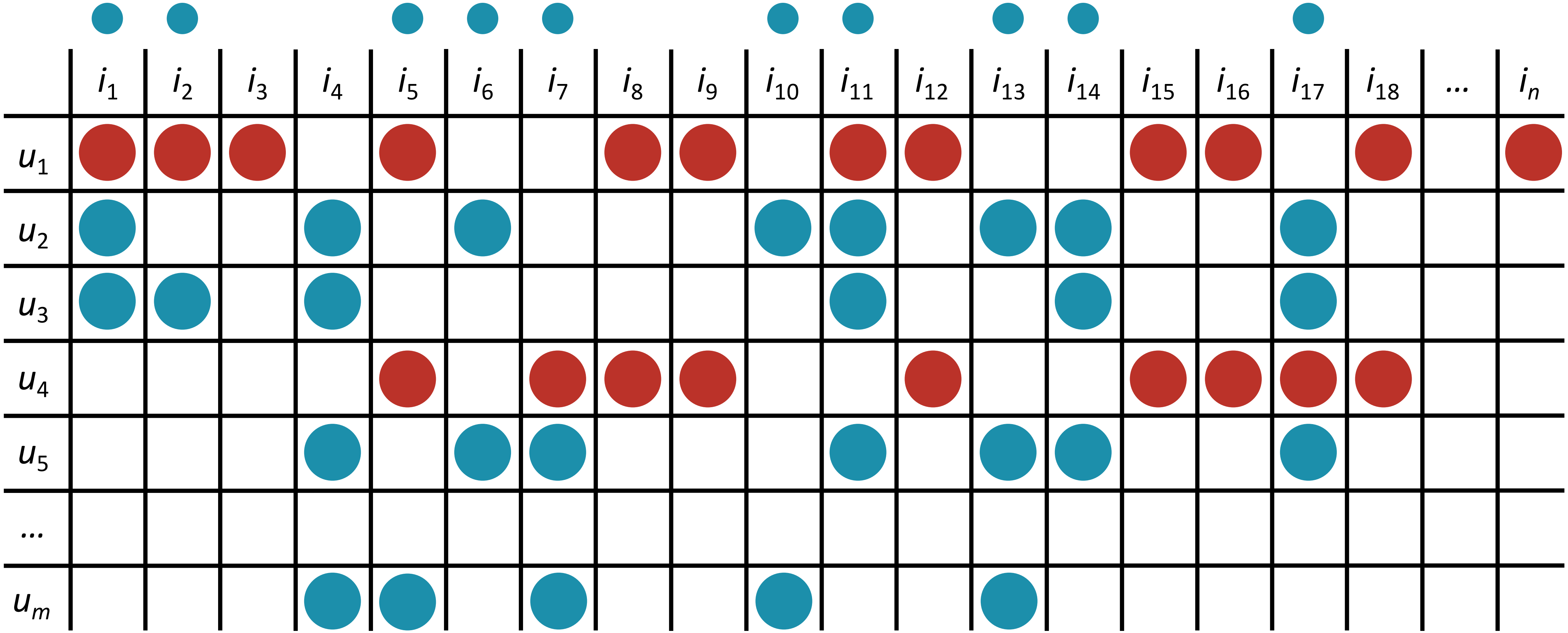
Item-based collaborative filtering

	i_1	i_2	i_3	i_4
u_1	1.3	-0.7	-0.7	?
u_2	1.0			-1.0
u_3		-0.3	0.7	-0.3

Item-based collaborative filtering



Item-based collaborative filtering



Item representation stability

Item vectors are typically **less sparse**

- Successful sites have many more users than items
- Often an item can be consumed by multiple users

Item vectors are considerably **more stable**

- A new feedback shouldn’t scratch item similarities

Precomputed item similarities are more durable

Model building

Stability allows for model-based computation

- Precompute similarities for all item pairs
- Model contains list of neighbors for each item

Still a costly operation

- Naively: $O(n^2m)$
- But can be performed offline

Model storage

No need to keep all neighbors

- More neighbors → better coverage
- Less neighbors → better efficiency

Balance memory usage and accuracy

- Keep enough neighbors to recommend (typically, $k \ll x \ll n$)

Computing similarities

	i_1	i_2	i_3	i_4
u_1	1.3	-0.7	-0.7	?
u_2	1.0			-1.0
u_3		-0.3	0.7	-0.3

$$s_{\vec{i}_4 \vec{i}_1} = -0.58$$

$$s_{\vec{i}_4 \vec{i}_2} = +0.11$$

$$s_{\vec{i}_4 \vec{i}_3} = -0.20$$

Aggregating ratings

	i_1	i_2	i_3	i_4	
u_1	1.3	-0.7	-0.7	?	$\tilde{r}_{u_1 i_4}$
u_2	1.0			-1.0	
u_3		-0.3	0.7	-0.3	

Aggregating ratings

	i_1	i_2	i_3	i_4
u_1	1.3	-0.7	-0.7	?

$\tilde{r}_{u_1 i_4}$

$\tilde{r}_{u_1 i_1}$ $\tilde{r}_{u_1 i_2}$ $\tilde{r}_{u_1 i_3}$

$s_{\vec{i}_4 \vec{i}_1}$ $s_{\vec{i}_4 \vec{i}_2}$ $s_{\vec{i}_4 \vec{i}_3}$

Aggregating ratings

	i_1	i_2	i_3	i_4
u_1	1.3	-0.7	-0.7	?

$$\tilde{r}_{u_1 i_4} = \frac{\sum_{c=1}^{k} s_{\vec{i}_4 \vec{i}_c} \tilde{r}_{u_1 i_c}}{\sum_{c=1}^{k} \left| s_{\vec{i}_4 \vec{i}_c} \right|}$$

$$= \frac{(-0.58 \times 1.3) + (0.11 \times -0.7) + (-0.20 \times -0.7)}{|-0.58| + |0.11| + |-0.20|} = -0.77$$

Aggregating ratings

	i_1	i_2	i_3	i_4
u_1	1.3	-0.7	-0.7	?

$$\tilde{r}_{u_1 i_4} = -0.77$$

$$\begin{aligned} \hat{r}_{u_1 i_4} &= \tilde{r}_{u_1 i_4} + \bar{r}_{u_1} \\ &= -0.77 + 3.7 \\ &= 2.93 \end{aligned}$$

Flexible neighborhood selection

k items most similar to…

- … items rated by $\boldsymbol{u}$
- … items just viewed by $\boldsymbol{u}$
- … items added to $\boldsymbol{u}$'s basket

Memory-based collaborative filtering

How will user $\boldsymbol{u}$ like item $\boldsymbol{i}$?

- User-based: *how do users similar to* $\boldsymbol{u}$ *like* $\boldsymbol{i}$*?*
- Item-based: *how do* $\boldsymbol{u}$ *like items similar to* $\boldsymbol{i}$*?*

Key difference: **neighborhoods**

- User-based: **unstable**, hard to precompute
- Item-based: **stable**, easy to precompute

Summary

Item-based CF is **effective**

- More resilient to data sparsity

Item-based CF is **efficient**

- Stability allows neighborhood precomputation

Item-based CF is **flexible**

- Profile-, session-, basket-based neighborhood

References

Recommender Systems: An Introduction (Sec. 2.2)

Recommender Systems Handbook (Sec. 2.3)

Recommender Systems: The Textbook (Sec. 2.3)