UFMG UNIVERSIDADE FEDERAL DE MINAS GERAIS

Recommender Systems

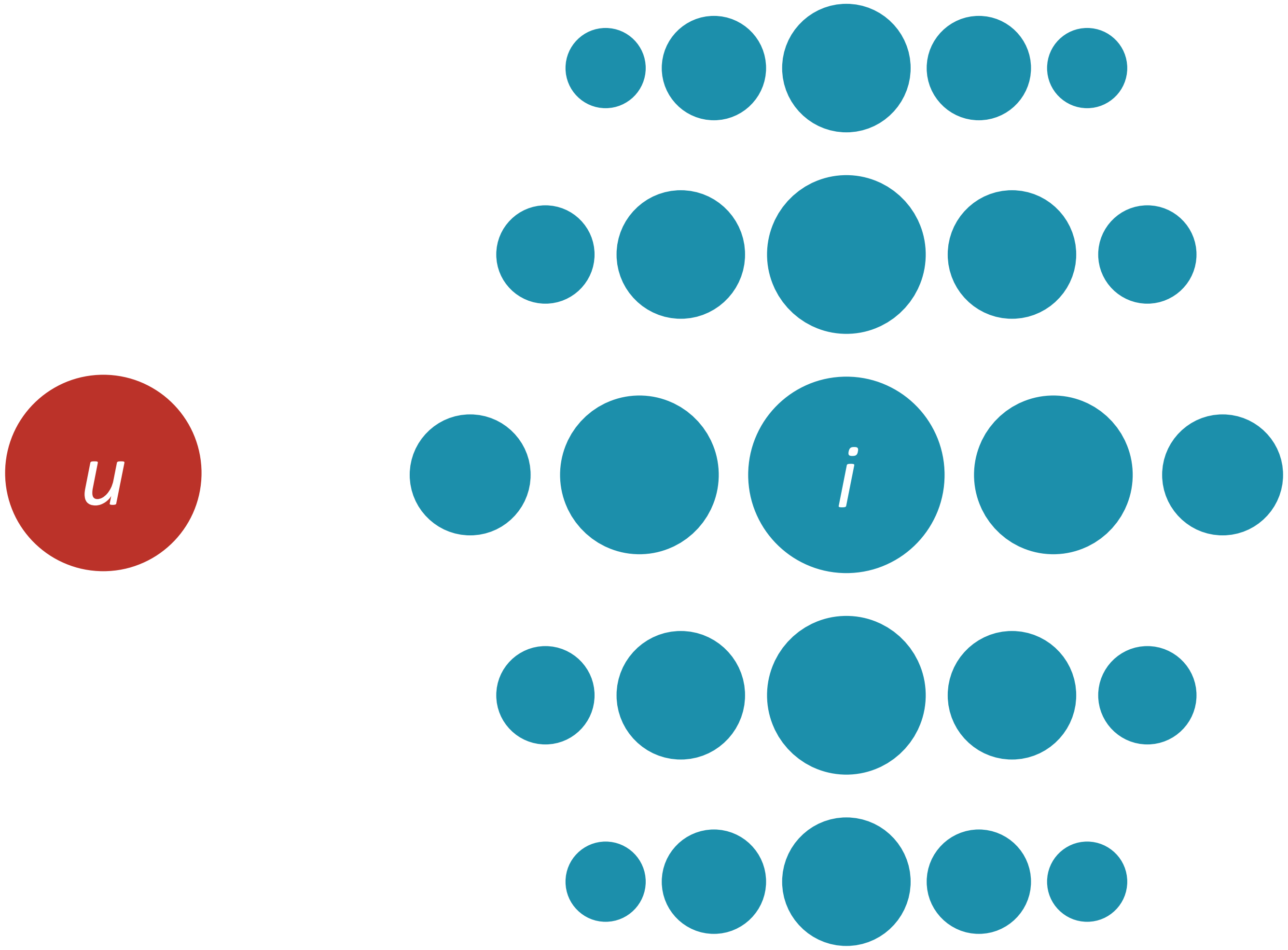
# User-Based Collaborative Filtering

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

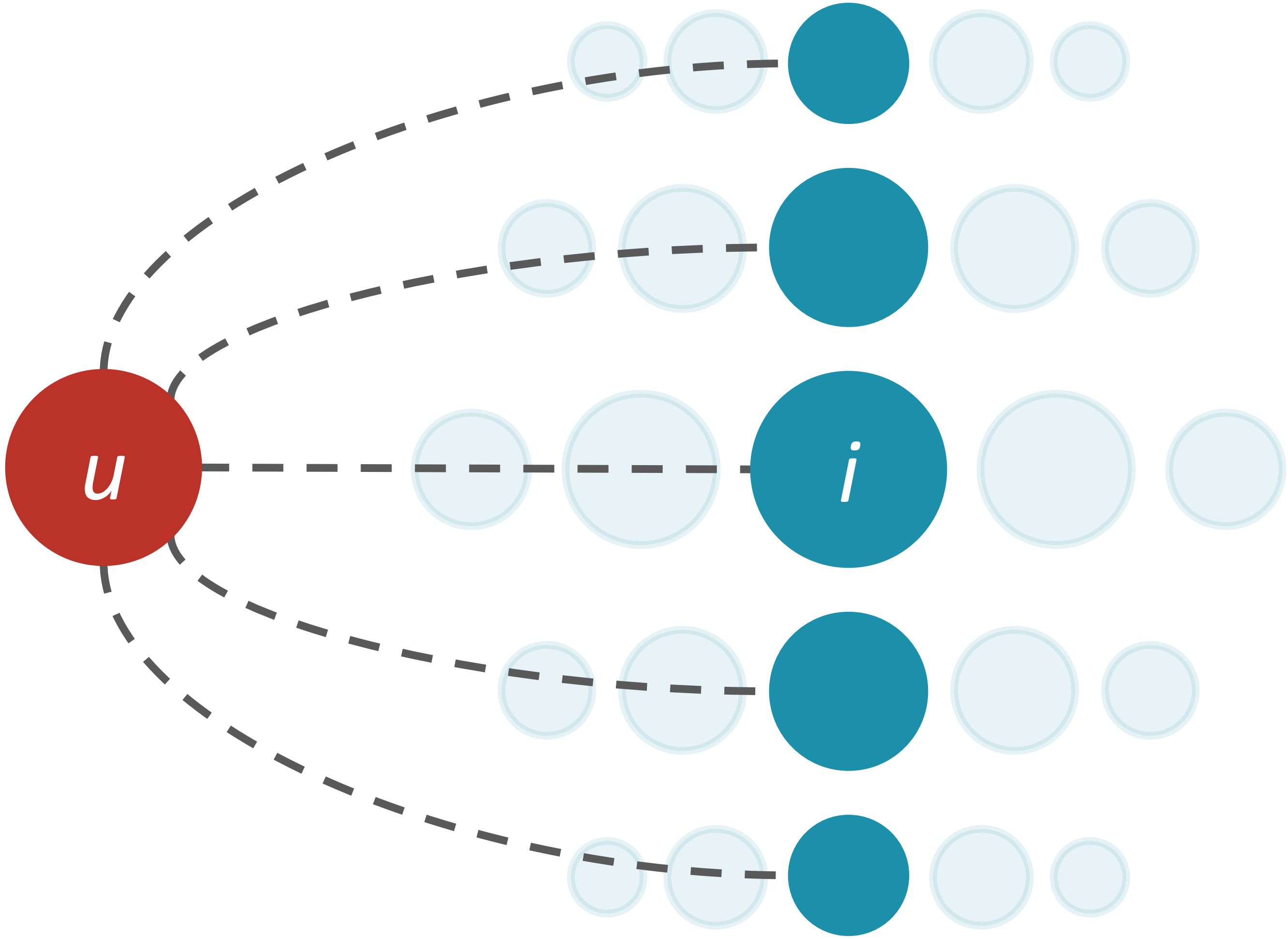
# The recommendation problem


$$f(u, i)$$

# The recommendation problem



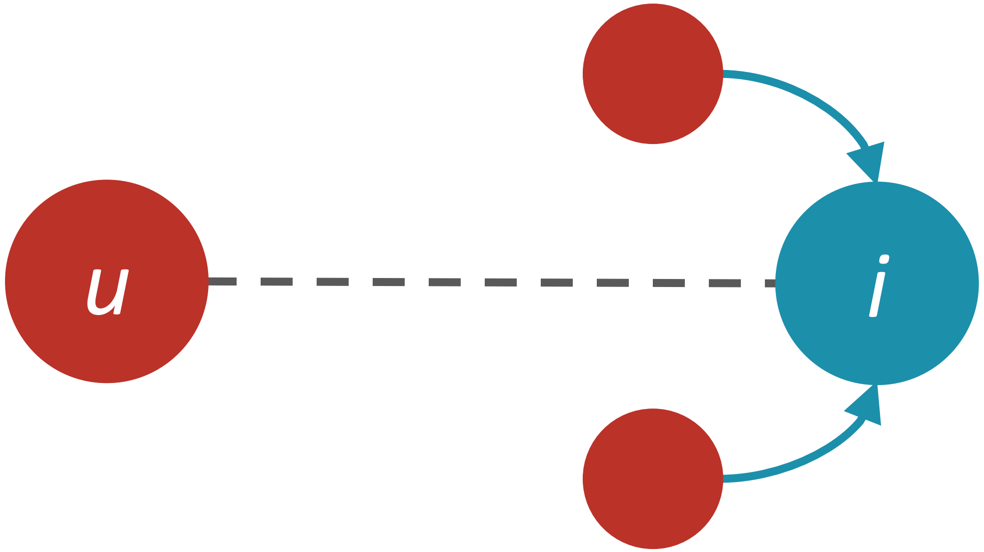
# The recommendation problem



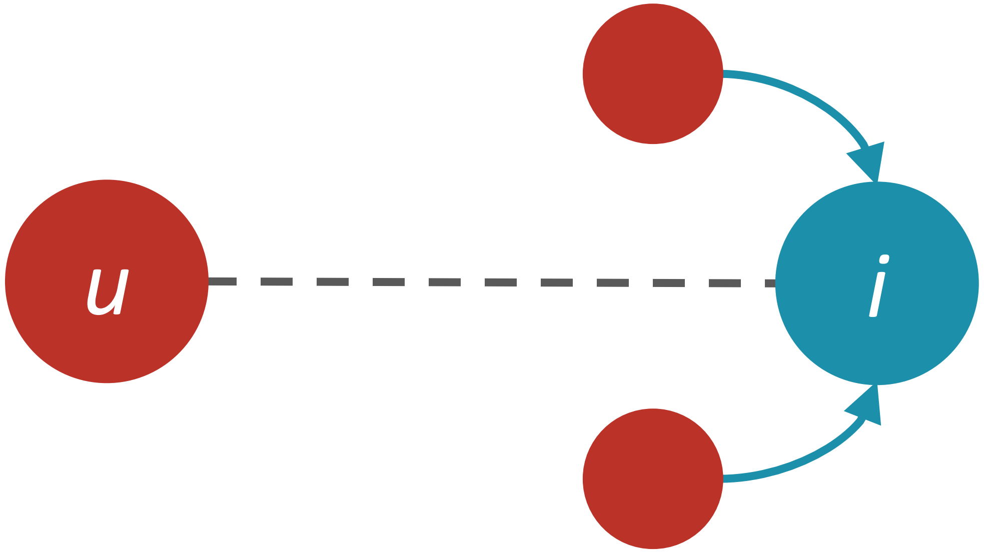
# The recommendation problem


$$f(u,i)$$

# Cold-start user



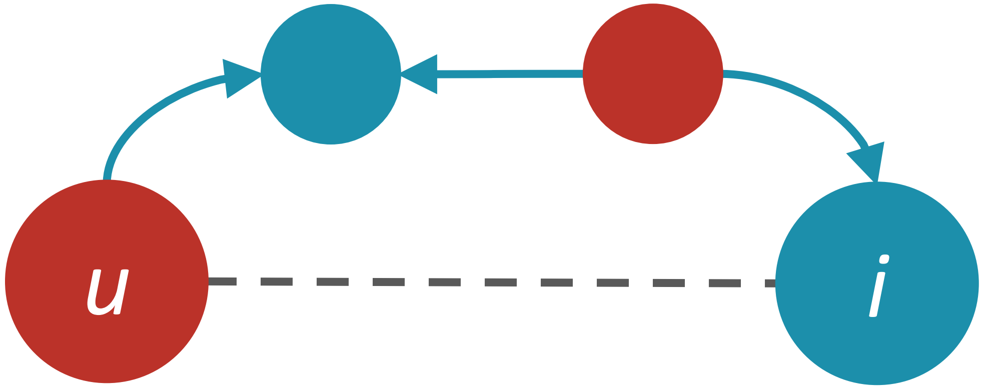
# Non-personalized recommendation




most recent
most popular
best rated

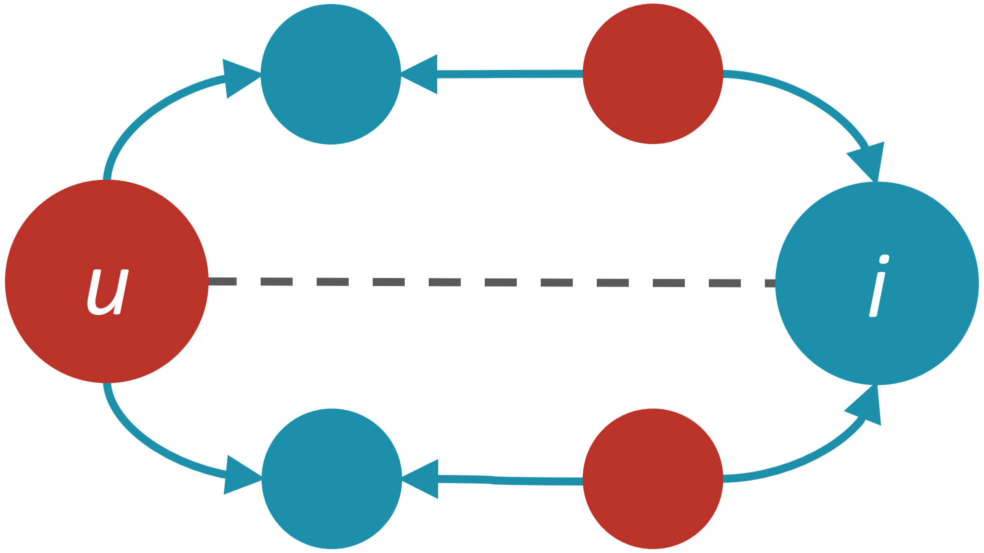
# What if we know the user?

# Personalized recommendation



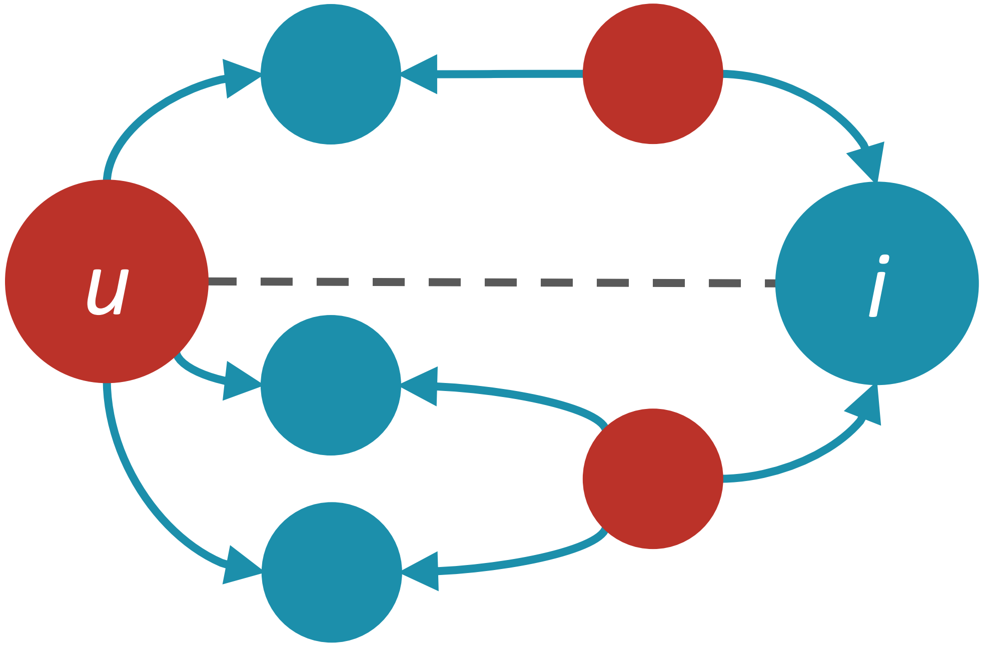

exploit **paths** via other users for **personalization**

# Personalized recommendation



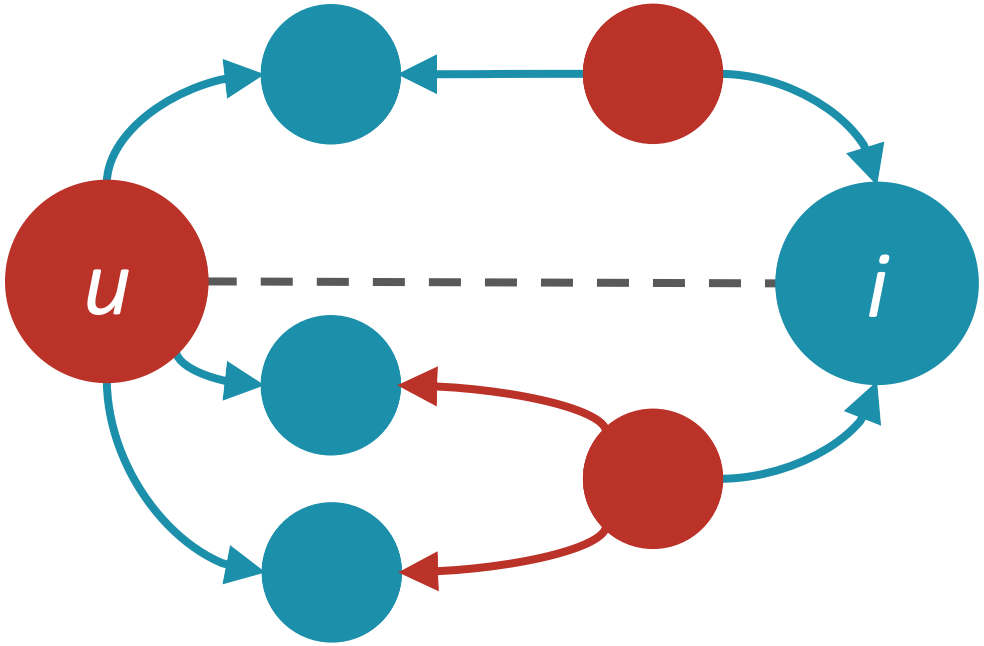

**more** users,
better signal?

# Personalized recommendation



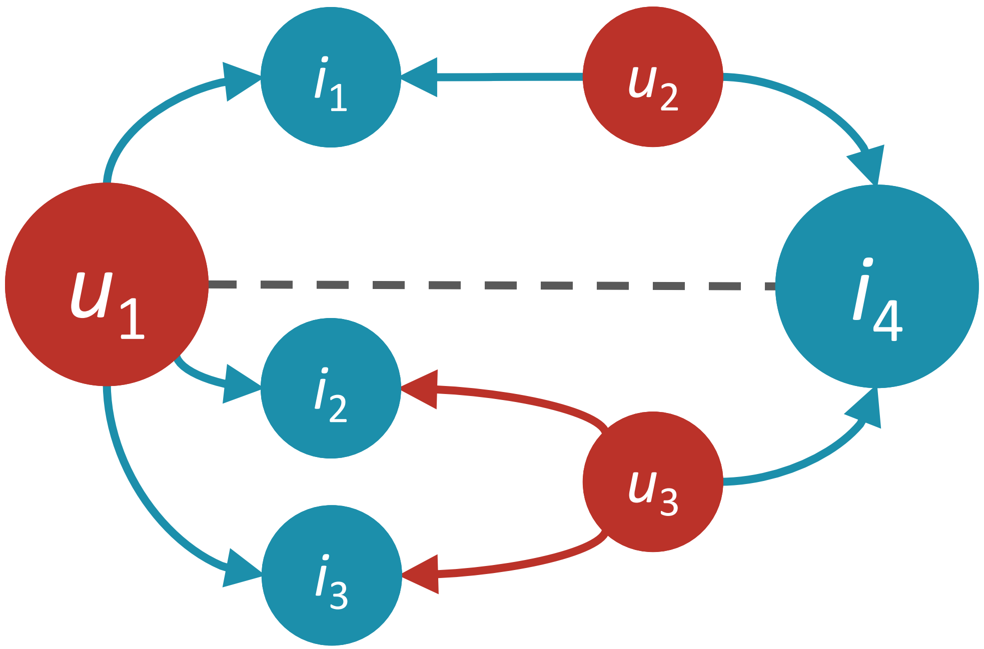

**coincident** users,
better signal?

# Personalized recommendation




**similar** users,
better signal!

# User-based collaborative filtering



# User-based collaborative filtering

| | $i_1$ | $i_2$ | $i_3$ | $i_4$ |
|---|---|---|---|---|
| $u_1$ | +1 | +1 | +1 | |
| $u_2$ | +1 | | | +1 |
| $u_3$ | | | | +1 |

**unary** feedback
(e.g., click)

# User-based collaborative filtering

| | $i_1$ | $i_2$ | $i_3$ | $i_4$ |
|---|---|---|---|---|
| $u_1$ | +1 | +1 | +1 | |
| $u_2$ | +1 | | | +1 |
| $u_3$ | | -1 | -1 | +1 |

**binary** feedback
(e.g., like / dislike)

# 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 |

**graded** feedback
(e.g., rate)

# Breaking it down

Normalizing ratings

Computing similarities

Selecting neighborhoods

Aggregating ratings

# Normalizing ratings

Users rate differently

- Some rate high, others low
- Some use more of the scale than others

Aggregation ignores these differences

- Normalization compensates for them

# Mean-centering normalization

| | $i_1$ | $i_2$ | $i_3$ | $i_4$ | |
|---|---|---|---|---|---|
| $u_1$ | 5 | 3 | 3 | | $\bar{r}_{u_1} = 3.7$ |
| $u_2$ | 5 | | | 3 | $\bar{r}_{u_2} = 4.0$ |
| $u_3$ | | 1 | 2 | 1 | $\bar{r}_{u_3} = 1.3$ |

# Mean-centering normalization

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

# Selecting neighborhoods

A few options

- All the neighbors
- Random neighbors
- All neighbors above a **similarity** threshold
- Top-$k$ neighbors ranked by **similarity**

# Computing similarities

In principle, any similarity would do

- Jaccard, Dice index
- Pearson correlation
- Cosine

Empirically, some perform better

# Pearson correlation

$$s_{\vec{u}\vec{v}} = \frac{cov(\vec{u}\vec{v})}{\sigma_{\vec{u}}\sigma_{\vec{v}}}$$

$$\approx \frac{\sum_{i \in I_{uv}}(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}}(r_{ui} - \bar{r}_u)^2}\sqrt{\sum_{i \in I_{uv}}(r_{vi} - \bar{r}_v)^2}}$$

- Built-in mean-centering normalization
- Computed over ratings in common ($I_{uv}$)

# What about little data?

Two users with one common rating

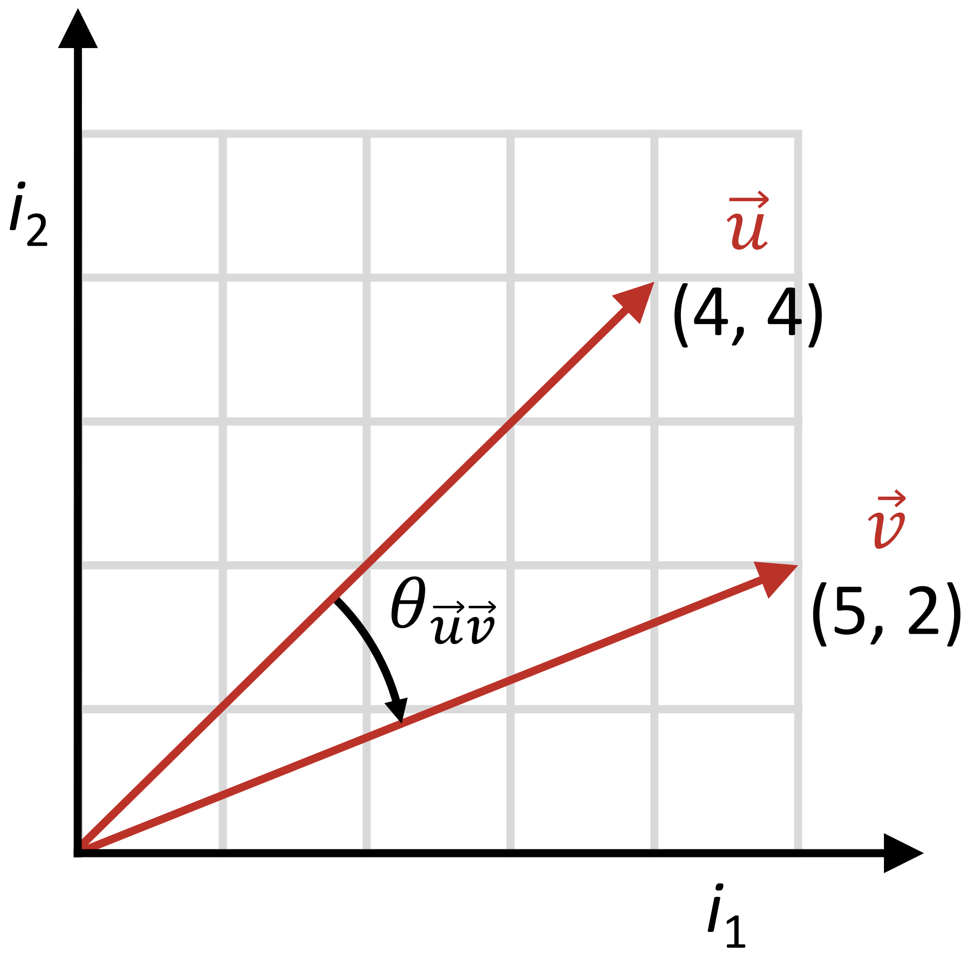
◦ $|I_{uv}| = 1 \rightarrow s_{\vec{u}\vec{v}} = 1$

Solution: weight similarity by confidence

◦ Simple approach: multiply by $min(|I_{uv}|, 50)/50$

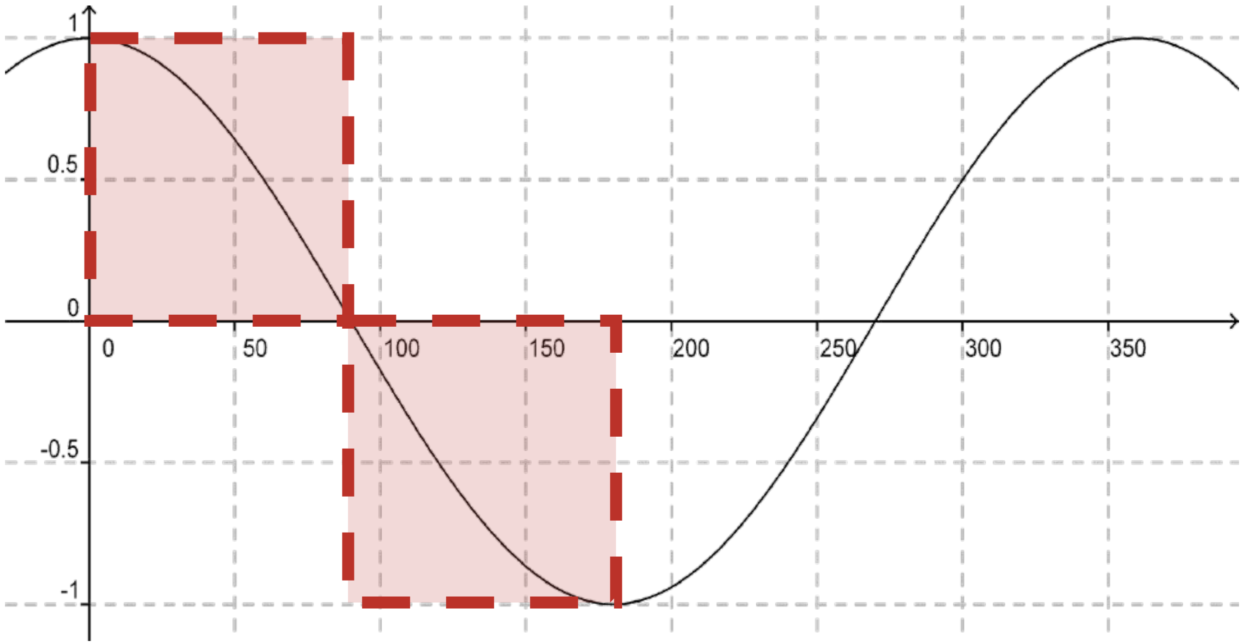
- $|I_{uv}| < 50$: similarity scaled down by $|I_{uv}|/50$
- $|I_{uv}| \geq 50$: similarity kept unscaled

# Cosine similarity



# From angles to cosines

Cosine is a monotonically decreasing function of the angle for the interval $[0^o, 180^o]$

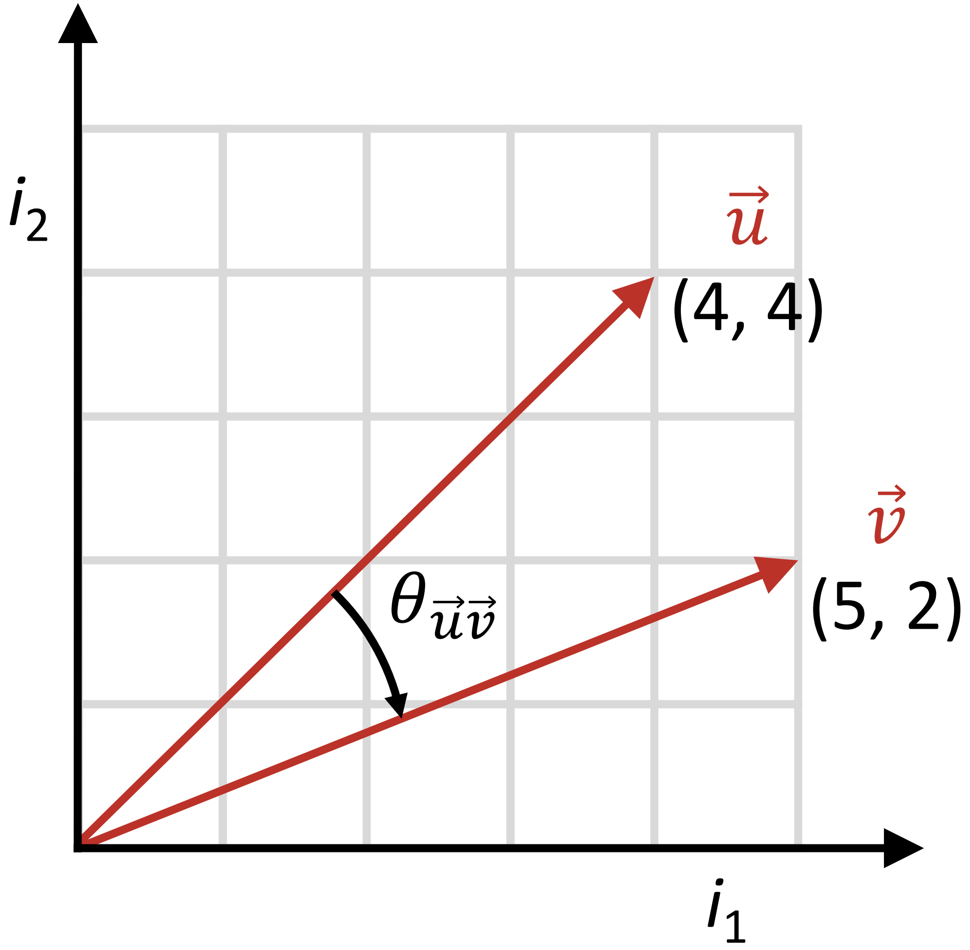



***smaller the angle, larger the cosine***

*vs.*

***larger the angle, smaller the cosine***

# Cosine similarity

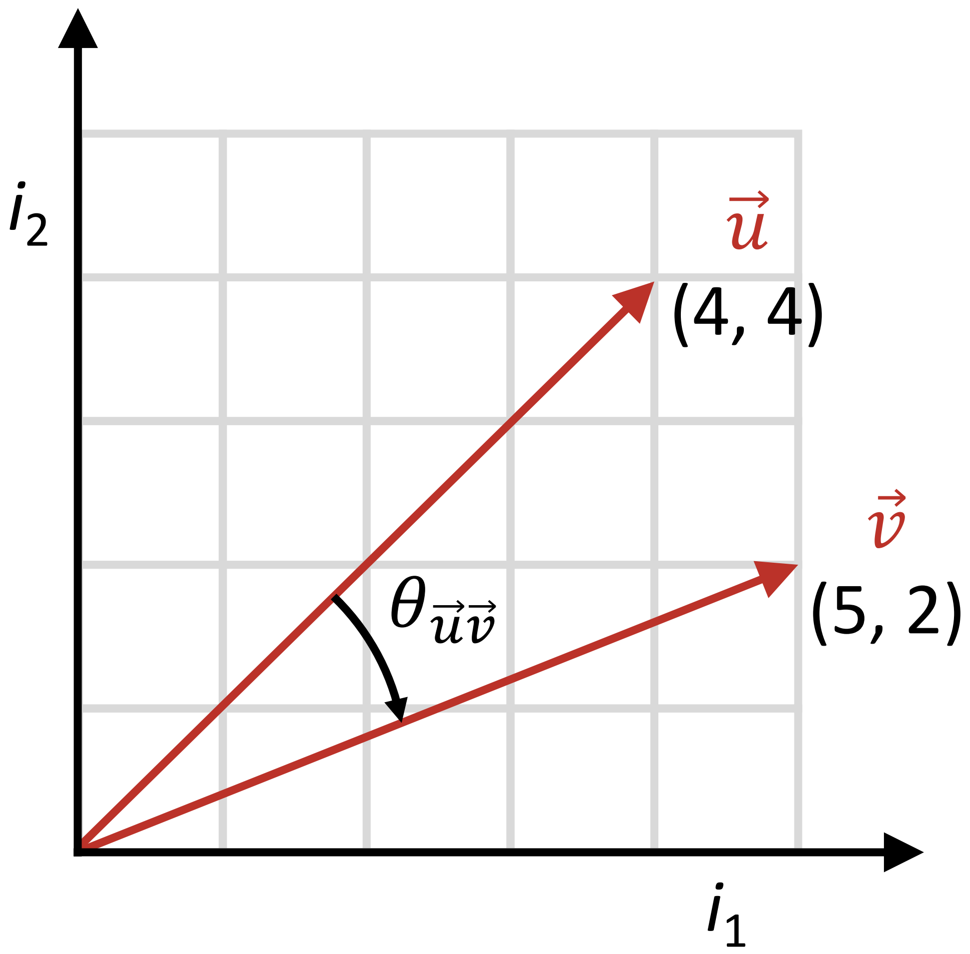



$$s_{\vec{u}\vec{v}} = \cos(\theta_{\vec{u}\vec{v}})$$

$$= \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|}$$

$$= \frac{\sum_{i \in I} r_{ui}\, r_{vi}}{\sqrt{\sum_{i \in I} r_{ui}^2}\, \sqrt{\sum_{i \in I} r_{vi}^2}}$$

# Cosine similarity




$$s_{\vec{u}\vec{v}} = \frac{4\times5 + 4\times2}{\sqrt{4^2 + 4^2}\,\sqrt{5^2 + 2^2}}$$

$$\approx 0.92$$

# Cosine similarity

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

$s_{\vec{u}_1\vec{u}_2} = +0.58$

$s_{\vec{u}_1\vec{u}_3} = -0.17$

# Cosine similarity

$$s_{\vec{u}\vec{v}} = \cos(\theta_{\vec{u}\vec{v}}) = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|} = \frac{\sum_{i\in I} r_{ui}\, r_{vi}}{\sqrt{\sum_{i\in I} r_{ui}^2}\sqrt{\sum_{i\in I} r_{vi}^2}}$$

- In general: $s_{\vec{u}\vec{v}} \in [-1,1]$
- With non-negative ratings: $s_{\vec{u}\vec{v}} \in [0,1]$
- With mean-centering (aka ***adjusted cosine***)
  - Equivalent to Pearson… ***almost!***

# Cosine vs. Pearson

Cosine has built-in significance weighting

- Similarity scaled by ratio

$$|I_u \cap I_v| \,/\, |I_u||I_v|$$

- Similar effect can be obtained by using **overall σ** instead of just **σ over common ratings** in Pearson

# How many neighbors?

In theory, the more the better…

- … if you have a good similarity metric
- Computational cost is also higher

In practice

- More neighbors → more noise
- Fewer neighbors → lower coverage

# Aggregating ratings

A few options

- Min / max / average / median rating
- Weighted average (by similarity)
- Supervised aggregation

Common practice

- Weighted average: simple and effective

# Aggregating ratings

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

$$\hat{r}_{u_1 i_4} - \bar{r}_{u_1} = \tilde{r}_{u_1 i_4}$$

$$r_{u_2 i_4} - \bar{r}_{u_2} = \tilde{r}_{u_2 i_4}$$

$$r_{u_3 i_4} - \bar{r}_{u_3} = \tilde{r}_{u_3 i_4}$$

# Aggregating ratings

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

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

$= \tilde{r}_{u_2 i_4} \quad s_{\vec{u}_1 \vec{u}_2} = +0.58$

$= \tilde{r}_{u_3 i_4} \quad s_{\vec{u}_1 \vec{u}_3} = -0.17$

# Aggregating ratings

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

$$\tilde{r}_{u_1 i_4} = \frac{\sum_{c=1}^{k} s_{\vec{u}_1 \vec{u}_c} \tilde{r}_{u_c i_4}}{\sum_{c=1}^{k} \left| s_{\vec{u}_1 \vec{u}_c} \right|}$$

$$= \frac{(0.58 \times -1.0) + (-0.17 \times -0.3)}{|-0.58| + |-0.17|}$$

$$= -0.71$$

# Aggregating ratings

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

$$\tilde{r}_{u_1 i_4} = -0.71$$

$$\begin{aligned}\hat{r}_{u_1 i_4} &= \tilde{r}_{u_1 i_4} + \bar{r}_{u_1} \\ &= -0.71 + 3.7 \\ &= 2.99\end{aligned}$$

# Suggested configuration

Reasonable starting point

- Top-$k$ neighbors ($k \approx 30$)
- Weighted averaging of scores
- Mean-centering normalization
- Cosine similarity

***Optimal configuration is application-dependent***

# 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?***

# Summary

User-based CF is simple and effective

- The oldest CF approach

Lots of configuration knobs

- Similarity functions, neighborhood selection, aggregation functions, rating normalization

Neighborhood selection is a bottleneck