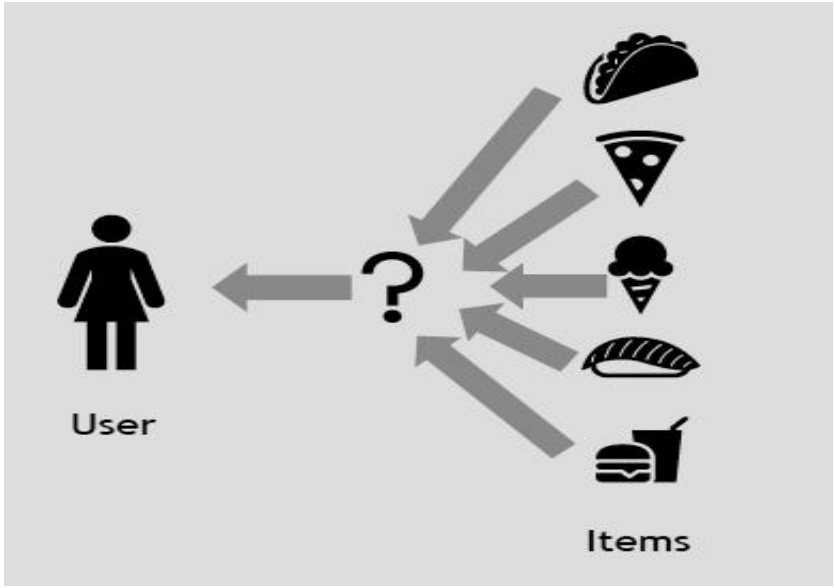


Section 6.2: Content-based Recommender Systems

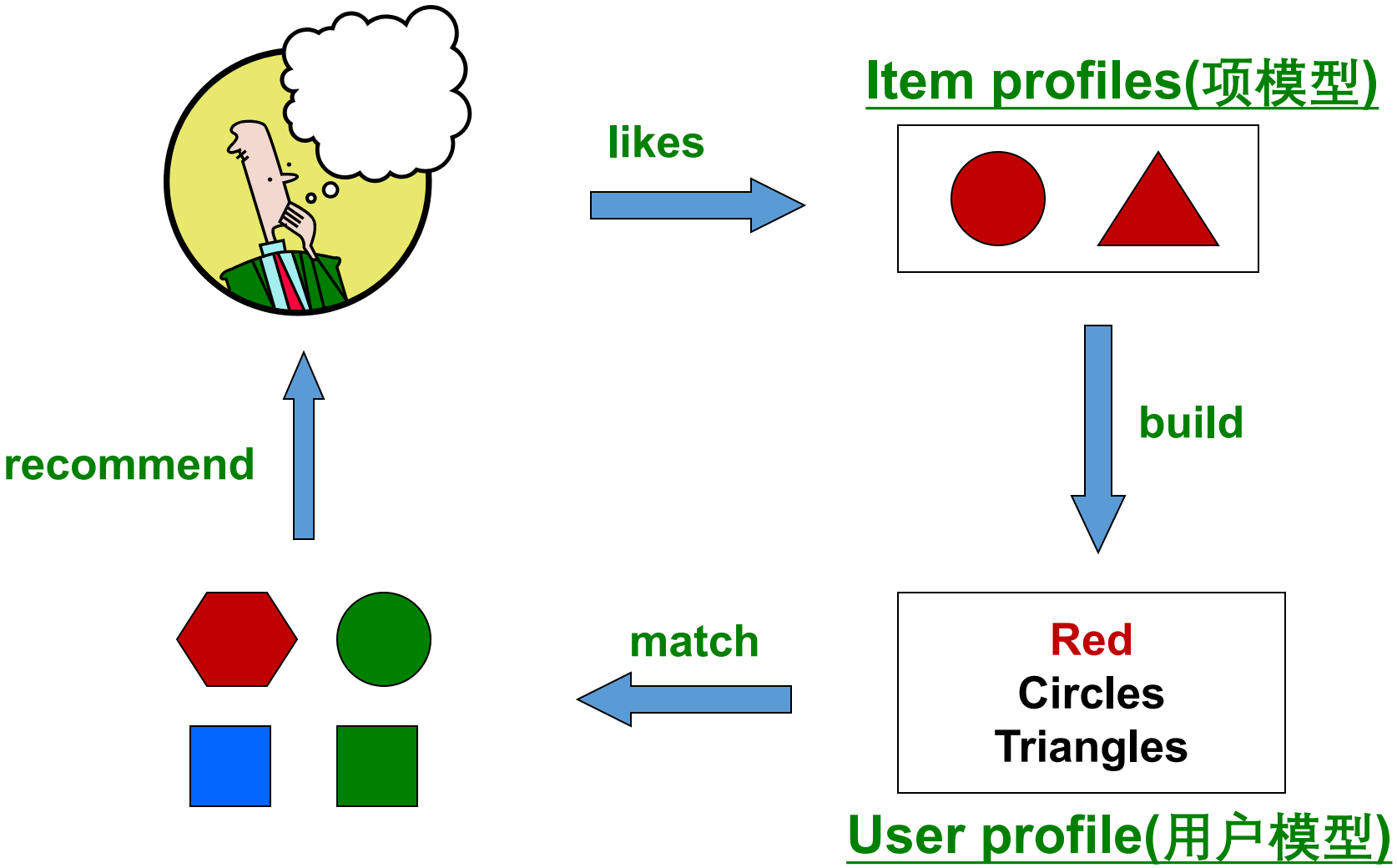
Content-based Recommendations

- **Main idea:** Recommend items to customer ***x*** similar to previous items rated highly by ***x***

Example:

- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, …
- **Websites, blogs, news**
  - Recommend other sites with "similar" content

Plan of Action



Item Profiles

- For each item, create an **item profile (项模型)**
- **Profile is a set (vector) of features**
  - **Movies:** author, title, actor, director,…
  - **Text:** Set of "important" words in document
- **How to pick important features?**
  - Usual heuristic from text mining is **TF-IDF** (Term frequency * Inverse Doc Frequency, 词项频率*逆文档频率, TF.IDF), 用于度量给定词语在少数文档中反复出现程度的形式化指标. 基于事实: 描述主题的词语往往都相对罕见.

TF-IDF

- **TF-IDF, including TF (Term frequency, 词项频率), IDF (Inverse Doc Frequency, 逆文档频率)**
  - N = total number of docs, 文档集中N篇文档
  - f_{ij} = frequency (or number of occurrences) of term (feature) i in doc (item) j, 词项i在文档j中出现的频率(次数)
- **TF**
  - $TF_{ij} = \frac{f_{ij}}{max_k f_{kj}}$
  - 词项i在文档j中的词项频率f_{ij}归一化结果, 归一化体现在f_{ij}通过除以同一个文档中出现最多的词项的频率.
  - TF_{ij}值最大为1, 其他都是小于1的分数

Note: we normalize TF to discount for "longer" documents

TF-IDF

- N = total number of docs, 文档集有N篇文档
- n_i= number of docs that mention term i, 文档集中包含词项i的文档数量

IDF

- $IDF_i = log_2 \frac{N}{n_i}$

TF-IDF score: $S_{ij} = TF_{ij} \times IDF_i$

Doc profile = set of words with highest **TF-IDF** scores, together with their scores, 具有最高**TF-IDF**得分的词项通常是刻画文档主题的最佳词项.

Example

- Suppose we have 2^{20}=1048576 documents, word w appears in $2^{10} = 1024$ of these documents.
  - 1) Consider a document j in which w appears 20 times, and that is also the maximum number of times in which any word appears. Then, what is the TF.IDF score for w in document j?
  - 2) Suppose word w appears 1 in document k, while the maximum number of occurrences of any word in this document is 20. What is the TF.IDF score for w in document k?
- Ans:
  - 1) TF_{wj}=20/20=1, IDF_w=$log_2 \frac{N}{n_i}$=10, so S_{wj}=10
  - 2) TF_{wj}=1/20, IDF_w=$log_2 \frac{N}{n_i}$=10, so S_{ij}=1/2

User Profiles

- **User profile (用户模型):** 我们不仅为项建立向量表示, 也需要将用户的偏好表示成同一空间下的向量. 一般可以通过效用矩阵上进行某些统计而来.

- 例: 项是电影, 通过片中演员构成的布尔向量来表示. 效用矩阵中如果用户看过某电影, 则对应元素为1, 否则为空白. 如果用户U看过的20%电影中都包含演员朱丽叶, 那么用户U的用户模型中朱丽叶对应的分量为0.2.

User Profiles

User profile (用户模型) possibilities:

- **Approach 1(平均法):** Weighted average of rated item profiles

$$w_u = \frac{1}{N}\sum_{i=1}^{N} w_i$$

N:用户历史喜欢物品的总数
w_i:用户喜欢的第i个物品的向量

- **Approach 2 (时间衰减):**在某些推荐场景下, 会考虑到时间效应, 用户的兴趣迁移比较快,其中的某个兴趣点可能会随着时间变化不断衰减, 因此在计算用户偏好特征的时候会增加时间因子

$$w_u = \frac{1}{N}\sum_{i=1}^{N} w_i \times \alpha^{diff_i}$$

α:小于1的参数, 该值越小时间衰减越快
$diff_i$:用户对物品i产生行为到现在的时间间隔

  - 时间衰减的算法很多，以上是一种经典的时间衰减的表达公式.

User Profiles

- **User profile (用户模型) possibilities:**
  - **Approach 3(Rocchio算法):**和平均法很类似，只是多了负反馈的部分，并分别给了两个调节权重实现灵活调节

$$w_u = \alpha \frac{1}{|I_r|} \sum_{w_j \in I_r} w_j - \beta \frac{1}{|I_{nr}|} \sum_{w_k \in In_r} w_k$$

I_r:用户喜欢的项集合
I_{nr}:用户不喜欢的项集合
w:某个项的特征向量
$\alpha\beta$:分别为正负反馈的权重

  - **Approach 4(机器学习算法)**:以上都是非机器学习算法，其实对用户画像也可以采用一些经典的机器学习算法，例如梯度提升树(GBDT)，对数几率回归(Logistic Regression)等

Prediction

- **Prediction heuristic**, 在项模型和用户模型的基础上, 可以采用不同的方式来估计用户喜欢某个项的程度:
  - **Approach 1(余弦相似度):** Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$, estimate $u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$

    If $u(\boldsymbol{x}, \boldsymbol{i})$ is close to 1, then that item is more recommended, and vice-versa

  - **Approach 2(BM25F算法):**来源于搜索领域, 相对复杂一些, 但效果比较好的概率检索模型(有兴趣的同学自学).

Learning a User Model

- 推荐系统除了使用项模型和用户模型进行基于内容的项推荐以外，也可以将推荐看为一个机器学习问题. 将给定数据看成训练集, 然后对用户建立一个分类器模型表示用户偏好, 预测用户对所有项的评分.
- The **training data** (i.e., user’s history) could be captured through explicit feedback (e.g., user rates items) or implicit observing of user’s interactions (e.g., user bought an item and later returned it is a sign of user doesn’t like the item)
  - Explicit method: perfect, but the amount collected could be limited
  - Implicit method: collect large amount of data, but contains noise

Learning a User Model

- Next, a number of classification learning algorithms (分类算法) are reviewed
  - Decision tree (决策树)
  - K nearest neighbours (KNN分类算法、K近邻算法)
  - ……
- The main goal of these classification learning algorithms is to learn a function that model the user’ s interests
  - Applying the function on a new item can give the probability that a user will like this item or a numeric value indicating the degree of interest in this item

Item Representation (1)

- Items stored in **a database table**

序号	餐厅名	风味	服务类型	费用
1001	Mike’ s Pizza	Italian	Counter	Low
1002	Chris’ s Café	French	Table	Medium
1003	Jacques Bistro	French	Table	High

- Structured data
  - Small number of attributes
  - Each item is described by the same set of attributes
  - Known set of values that the attributes may have

- Straightforward to work with
  - User’ s profile contains positive rating for 1001, 1002, 1003
  - Would the user be interested in say Oscars (French cuisine, table service)?

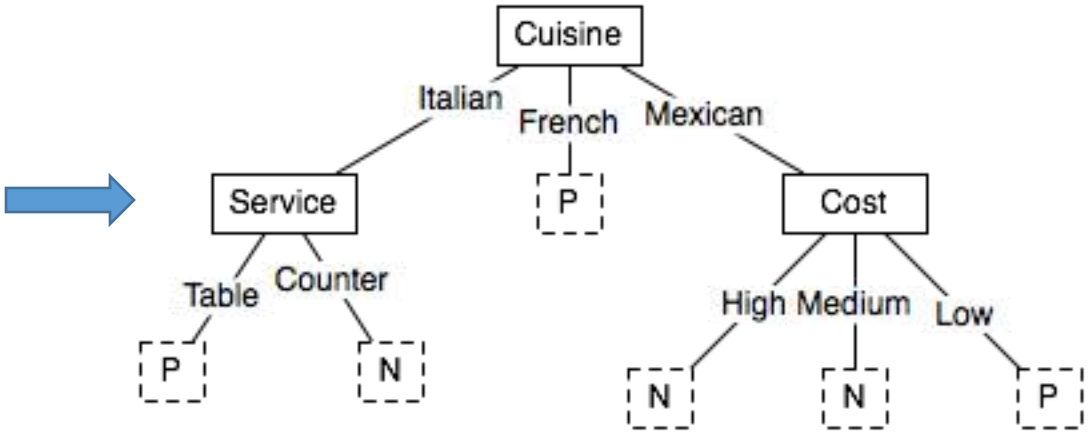
Item Representation (2)

- Information about item could also be **free text**; e.g., text description or review of the restaurant, or news articles
- Unstructured data
  - No attribute names with well-defined values
  - Natural language complexity
    - Same word with different meanings
    - Different words with same meaning
- Need to impose structure on free text before it can be used in recommendation algorithm

Decision Trees and Rule

- Given the history of user’s interests as training data, build a **decision tree (决策树)** which represents the **user’s profile of interest**

Cuisine	Service	Cost	Rating
Italian	Counter	Low	Negative
French	Table	Med	Positive
French	Counter	Low	Positive
…	…	…	…



- Will the user like an inexpensive Mexican restaurant? Yes!

Decision Trees and Rule

- Well-suited for structured data
- In unstructured data, the number of attributes becomes too enormous and consequently, the tree becomes too large to provide sufficient performance

- RIPPER算法，一种修剪后的决策树算法: A rule induction algorithm based on the same principles but provide better performance in classifying text (基于规则的分类器)
  - 有兴趣的同学可以课外自学该算法

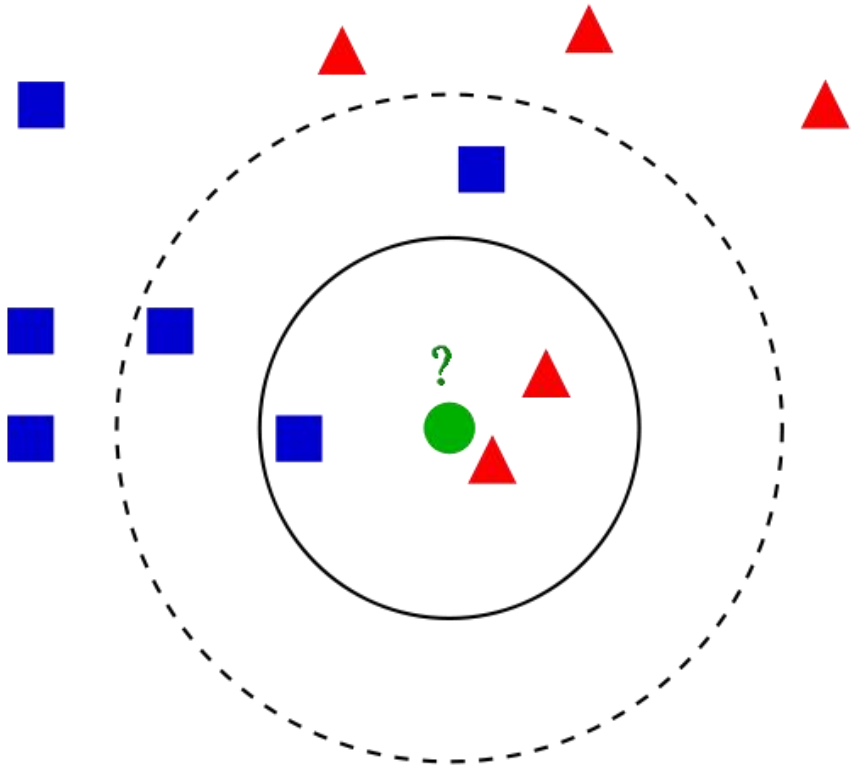
Nearest Neighbour Methods

- Simply store all the training data in memory
- To classify a new item, compare it to all stored items using a similarity function and determine the "nearest neighbour" or the ***k* nearest neighbours (KNN分类算法、K近邻算法).**
- The class or numeric score of the previously unseen item can then be derived from the class of the nearest neighbour.

Example: Nearest Neighbour Methods

- ● unseen item needed to be classified
- ■ positive rated items
- ▲ negative rated items

- Then, $k = 3$: negative
- $k = 5$: positive



Nearest Neighbour Methods

- The similarity function in KNN depends on the type of data:
  - **Structured data**: Euclidean distance metric, Minkowski distance, Mahalanobis distance,…
  - **Unstructured data** (i.e., free text): cosine similarity function,…

Similarity/Dissimilarity for Simple Attributes

- p and q are the attribute values for two data objects.

Attribute Type	Similarity （相似度）	Dissimilarity（相异度）
Nominal(二进制)	$s = \begin{cases} 1, & if\ p = q \\ 0, & if p \neq q \end{cases}$	$d = \begin{cases} 0, & if\ p = q \\ 1, & if p \neq q \end{cases}$
Ordinal(序列)	$s = 1 - \frac{\lvert p - q \rvert}{n - 1}$ (values mapped to integers 0 to $n - 1$, where n is the number of values)	$d = \frac{\lvert p - q \rvert}{n - 1}$
Interval or Ratio (比率)	$s = -d, s = \frac{1}{1 + d}\ or\ s = 1 - \frac{d - min_d}{max_d - min_d}$	$d = \lvert p - q \rvert$

Euclidean Distance

- Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects p and q.

- Standardization is necessary, if scales differ.

Minkowski Distance

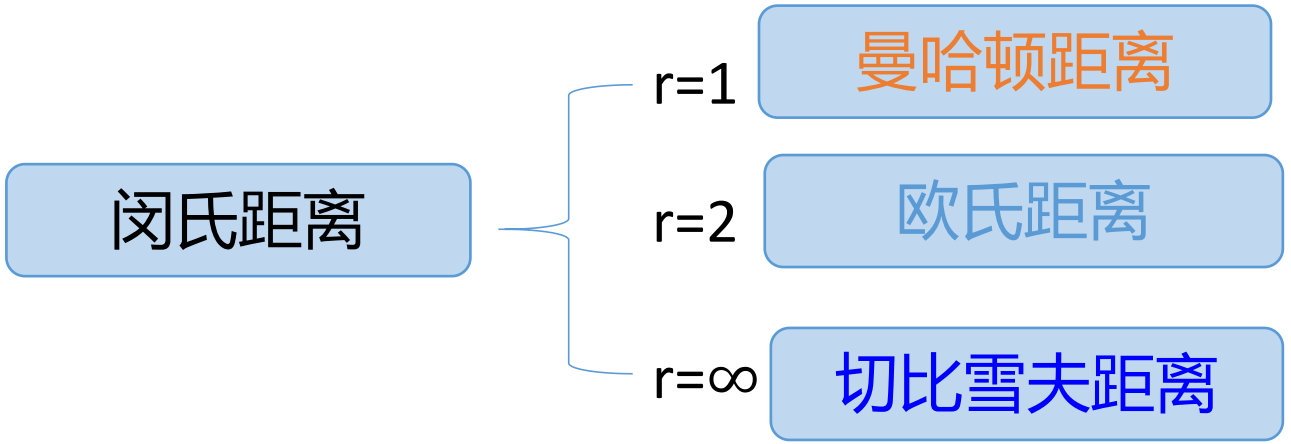
- Minkowski Distance (闵可夫斯基距，也称闵氏距离): a generalization of Euclidean Distance

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

where r is a parameter, n is the number of dimensions (attributes) and pk and qk are, respectively, the kth attributes (components) or data objects p and q.

$$Euclidean\ dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Minkowski Distance



$$dist = \left(\sum_{k=1}^{n} |p_k - q_k|^r\right)^{\frac{1}{r}}$$

- r = 1: City block (Manhattan, taxicab, L_1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors
- r = 2: Euclidean distance
- $r \to \infty$: "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component of the vectors

Examples: Minkowski Distance

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

$$dist = \left(\sum_{k=1}^{n} | p_k - q_k |^r \right)^{\frac{1}{r}}$$

r=1:

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

r=2:

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

r=∞:

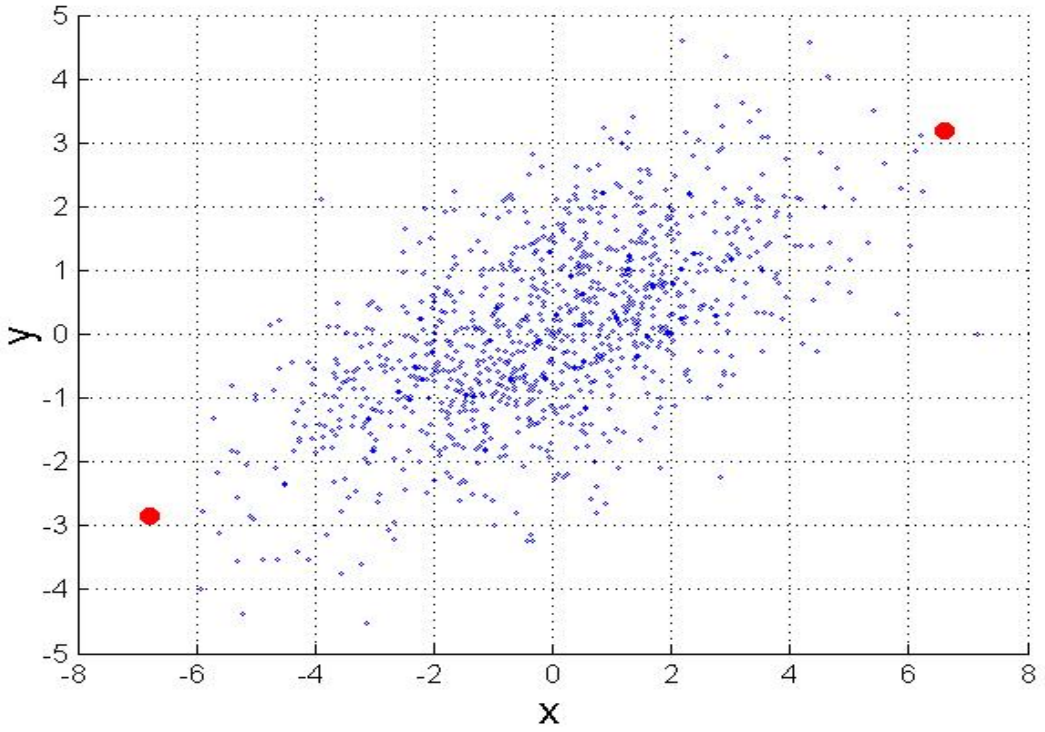
L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Distance Matrix

Mahalanobis Distance

- **Mahalanobis Distance (马氏距离):** is normalized Euclidean distance from centroid(欧氏距离的修正)

$$mahalanobis(p,q) = (p-q)\Sigma^{-1}(p-q)^T$$



Σ is the covariance matrix of the input data *X*

$$\Sigma_{j,k} = \frac{1}{n-1}\sum_{i=1}^{n}(X_{ij}-\bar{X}_j)(X_{ik}-\bar{X}_k)$$

For red points, the Euclidean distance is 14.7, Mahalanobis distance is 6.

Common Properties of a Distance

- Distances, such as the Euclidean distance, have some well known properties.
  - Positive definiteness: **d(p, q) ≥ 0** for all p and q, and d(p, q) = 0 only if p = q
  - Symmetry: **d(p, q) = d(q, p)** for all p and q.
  - Triangle Inequality: **d(p, r) ≤ d(p, q) + d(q, r)** for all points p, q, and r. where d(p, q) is the distance (dissimilarity) between points (data objects), p and q.
- A distance that satisfies these properties is a metric, e.g., Euclidean distance metric

Euclidean Distance Metric

Item	Attr. X	Attr. Y	Attr. Z
A	X_A	Y_A	Z_A
B	X_B	Y_B	Z_B

- Distance between A and B

$$d(A,B)=\sqrt{(x_A-x_B)^2+(y_A-y_B)^2+(z_A-z_B)^2}$$

- Attributes which are not measured quantitatively need to be labeled by numbers representing their categories
  - Cuisine attribute: 1=Frech, 2=Italian, 3=Mexican.

Cosine Similarity Function

- Vector space model
  - An item or a document d is represented as a vector
  - $w_{t,d}$ is the tf*idf weight of a term t in a document d

$$\mathbf{v}_d = \left[w_{1,d}, w_{2,d}, \ldots, w_{N,d}\right]^T$$

- **Cosine Similarity (余弦相似):** The similarity between two items can then be computed by the cosine of the angle between two vectors

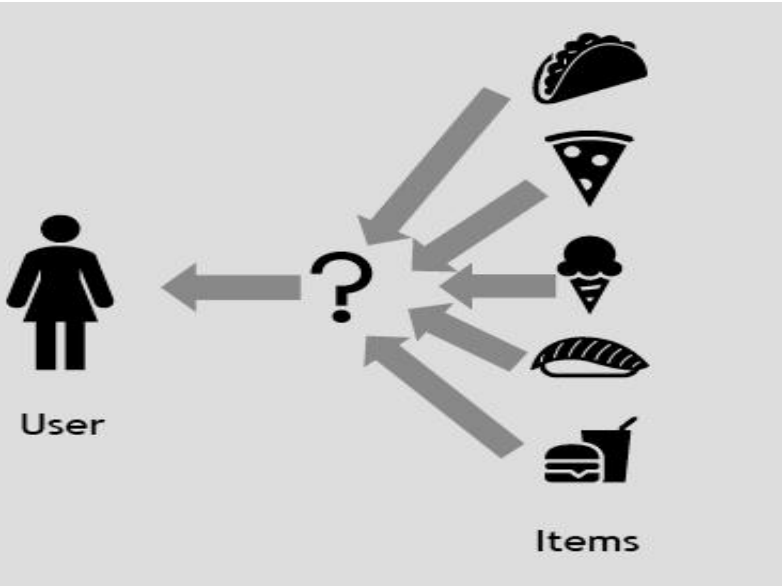
$$\cos\theta = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|}$$

Pros: Content-based Approach

- **+: No need for data on other users**
  - No cold-start or sparsity problems
- **+: Able to recommend to users with unique tastes**
- **+: Able to recommend new & unpopular items**
  - No first-rater problem
- **+: Able to provide explanations**
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended

Cons: Content-based Approach

- **–: Finding the appropriate features is hard**
  - E.g., images, movies, music
- **–: Recommendations for new users**
  - **How to build a user profile?**
- **–: Overspecialization**
  - Never recommends items outside user’ s content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users



Section 6.3: Collaborative Filtering

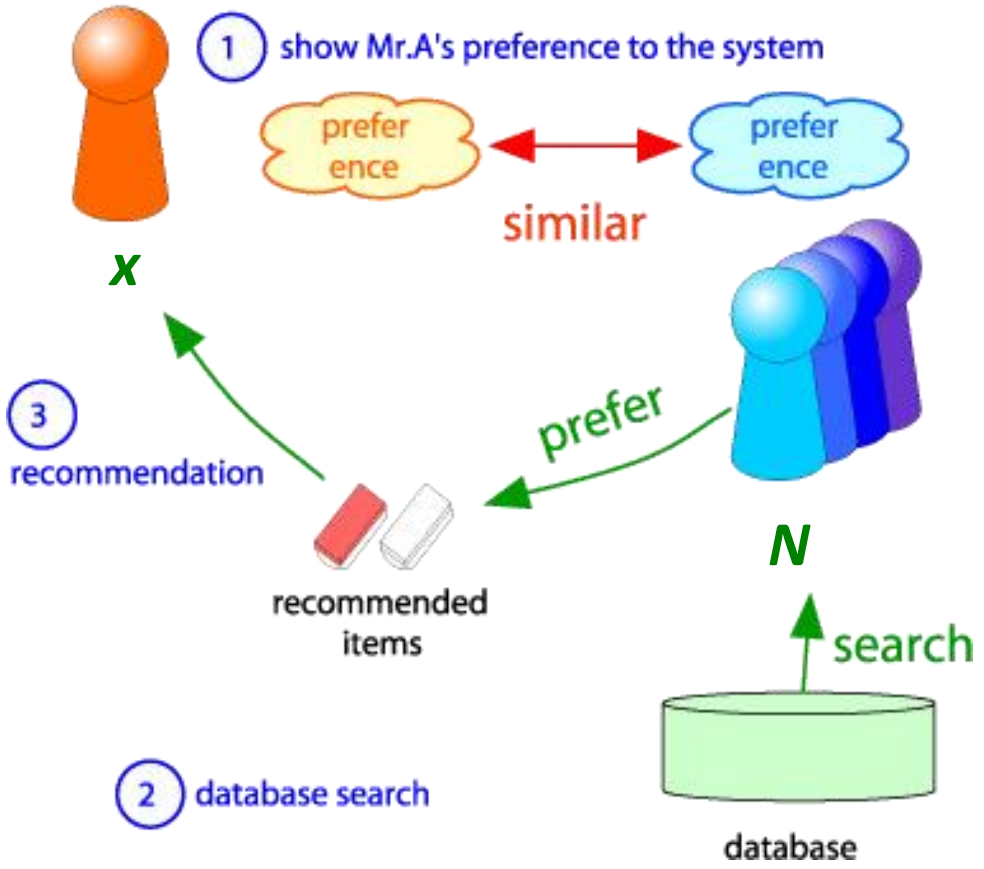
Harnessing quality judgments of other users

Collaborative Filtering

- Consider user ***x***

- Find set ***N*** of other users whose ratings are "**similar**" to ***x***' s ratings

- Estimate ***x***' s ratings based on ratings of users in ***N***



Finding "Similar" Users

r_x = [*, _, _, *, ***]
r_y = [*, _, **, **, _]

華中科技大學
计算机科学与技术学院
School of Computer Science &Technology，HUST

- Let r_x be the vector of user x's ratings
- **Jaccard similarity measure**
  - **Problem:** Ignores the value of the rating

r_x, r_y as sets:
r_x = {1, 4, 5}
r_y = {1, 3, 4}

- **Cosine similarity measure**
  - $\text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||}$
  - **Problem:** Treats missing ratings as "negative"

r_x, r_y as points:
r_x = {1, 0, 0, 1, 3}
r_y = {1, 0, 2, 2, 0}

- **Pearson correlation coefficient, 皮尔逊相关系数**
  - S_{xy} = items rated by both users x and y

$$sim(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

$\overline{r_x}$, $\overline{r_y}$ … avg. rating of **x**, **y**

Similarity Example

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

HP: 哈利波特
TW: 暮光之城
SW: 星球大战

- **Intuitively we want: sim(*A*, *B*) > sim(*A*, *C*)**
- **Jaccard similarity:** 1/5 < 2/4
- **Cosine similarity:** 0.386 > 0.322
  - Considers missing ratings as "negative"
  - **Solution: subtract the (row) mean**

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	2/3			5/3	−7/3		
B	1/3	1/3	−2/3				
C				−5/3	1/3	4/3	
D		0					0

sim(A,B) vs. sim(A,C):
0.092 > -0.559

Notice cosine sim. is correlation when data is centered at 0

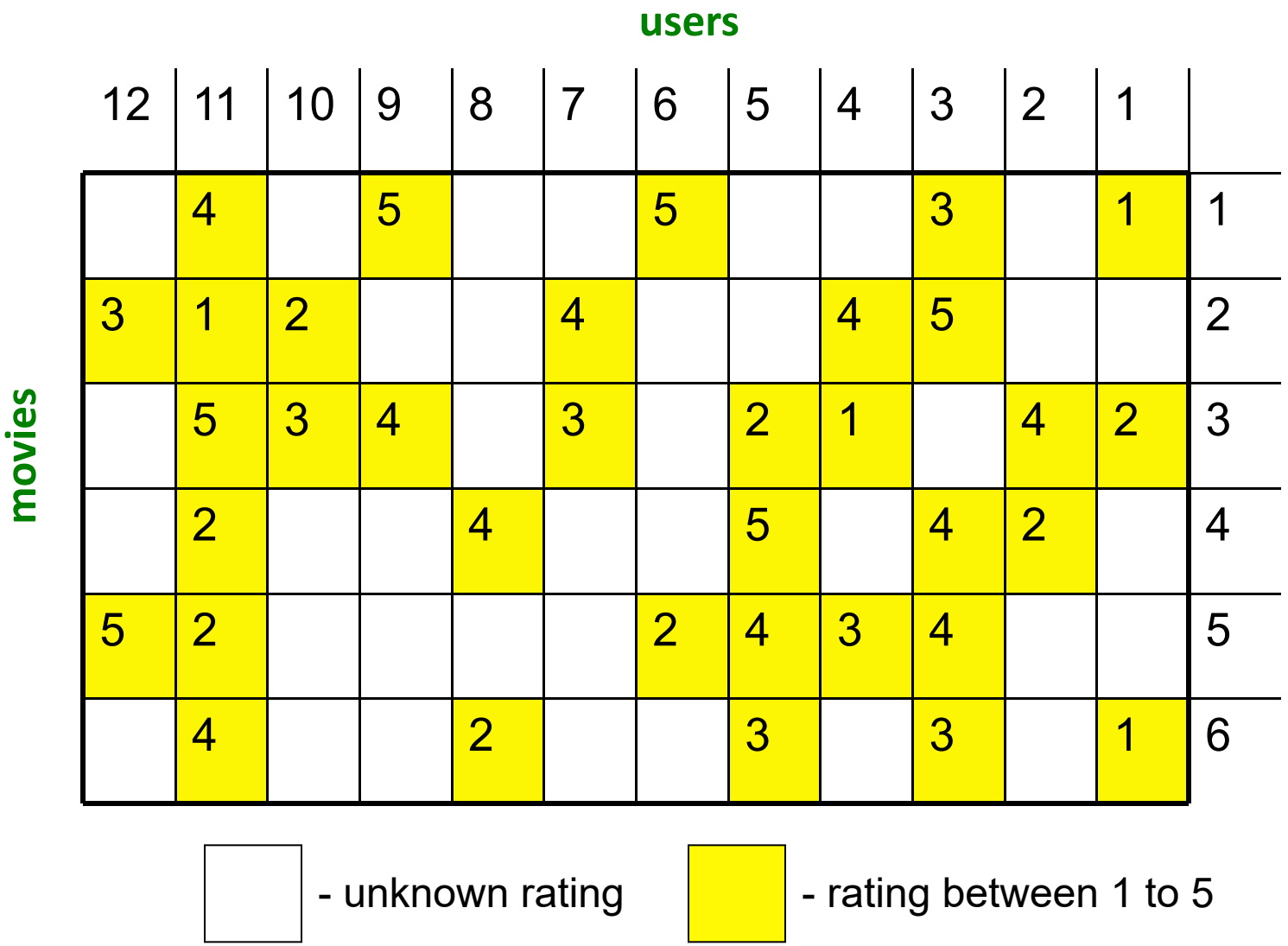
Rating Predictions

From similarity metric to recommendations:

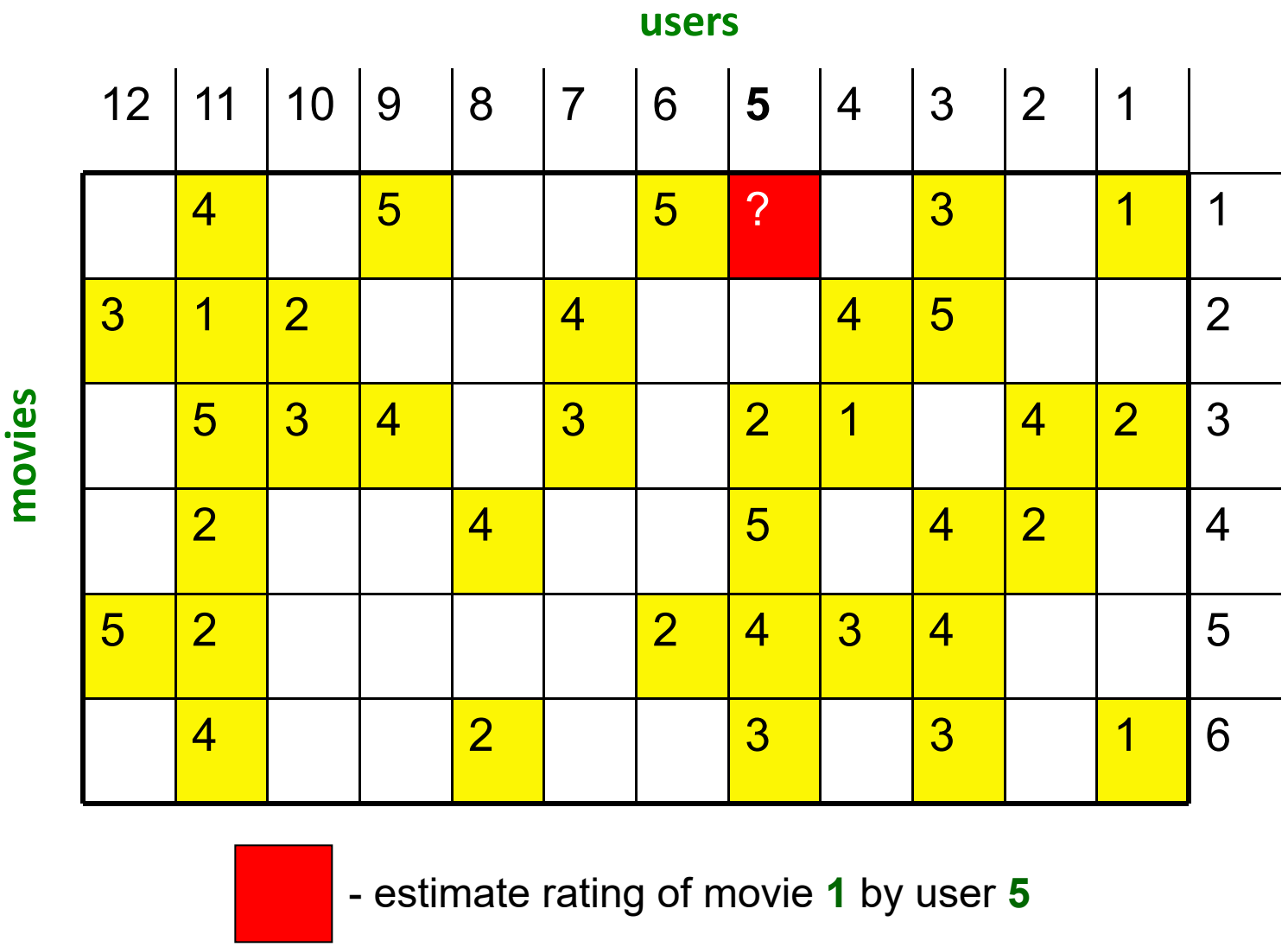
- Let r_x be the vector of user x's ratings
- Let N be the set of k users most similar to x who have rated item i (i.e., user-user CF)
- **Prediction for item *i* of *user x*:**
  - $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
  - $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

  Shorthand: $s_{xy} = sim(x, y)$
  - **Many other tricks possible…**

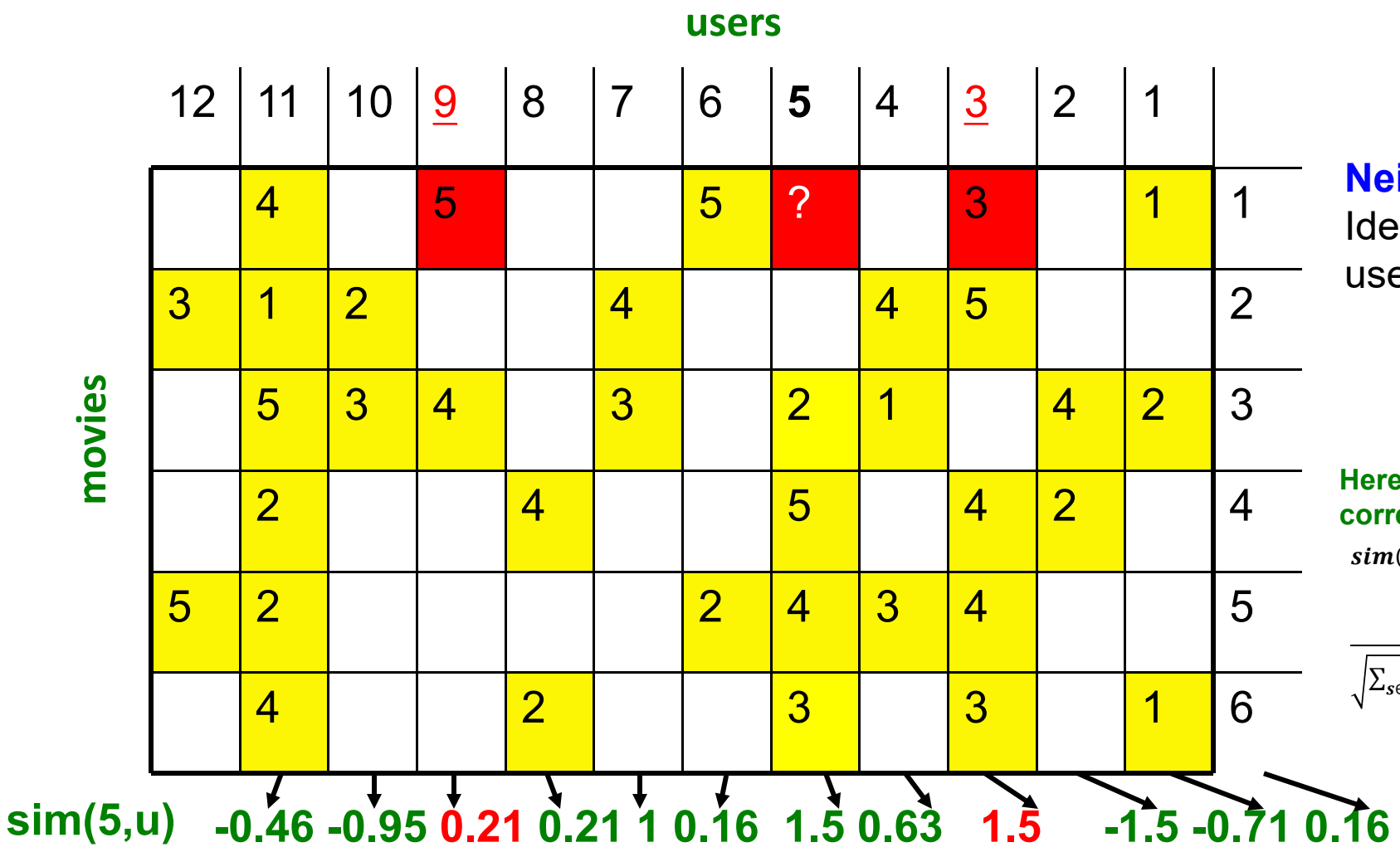
User-User CF (|N|=2)



User-User CF (|N|=2)



User-User CF (|N|=2)



Neighbor selection:
Identify users (N=2, so 2 users) similar to user 5

Here we use Pearson correlation as similarity

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

Note N should be the most similar to user 5 who have rated movie 1, so 0.21 and 1.5

User-User CF (|N|=2)

users

movies	12	11	10	9	8	7	6	**5**	4	3	2	1	
		4		5			5	?		3		1	1
	3	1	2			4			4	5			2
		5	3	4		3		2	1		4	2	3
		2			4			5		4	2		4
	5	2					2	4	3	4			5
		4			2			3		3		1	6

So user 3 and user 9

similarity weights: $s_{5,3}$=0.21, $s_{5,9}$=1.5

User-User CF (|N|=2)

users

12	11	10	9	8	7	6	**5**	4	3	2	1	
	4		5			5	4.75		3		1	1
3	1	2			4			4	5			2
	5	3	4		3		2	1		4	2	3
	2			4			5		4	2		4
5	2					2	4	3	4			5
	4			2			3		3		1	6

movies

Predict by taking weighted average:

$r_{1.5}$ = (0.21*3 + 1.5*5) / (0.21+1.5) =4.75

$$r_{ix} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{jx}}{\sum s_{ij}}$$