Big Data Computing

Master's Degree in Computer Science

2025-2026

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Recommendation Strategies

3 approaches to recommender systems

- Content-based filtering
- Collaborative filtering
  - Neighborhood-based (a.k.a. memory-based)
  - Latent-factor-based (a.k.a. model-based)
  - Hybrid (memory-model-based)
- Hybrid

COLLABORATIVE FILTERING

Collaborative Filtering (CF)

Idea

Recommend items to user u based on preferences of other users similar to u

Collaborative Filtering (CF)

Idea

Recommend items to user u based on preferences of other users similar to u

Core concept:

User-to-User or Item-to-Item similarity

Collaborative Filtering (CF)

Idea

Recommend items to user u based on preferences of other users similar to u

Core concept:
User-to-User or Item-to-Item similarity

No need for explicit creation of user/item profiles

Collaborative Filtering: Approaches

3 main approaches to collaborative filtering

Collaborative Filtering: Approaches

3 main approaches to collaborative filtering

Neighborhood-based
(a.k.a. memory-based)

Collaborative Filtering: Approaches

3 main approaches to collaborative filtering

Neighborhood-based
(a.k.a. memory-based)

Latent-factor-based
(a.k.a. model-based)

Collaborative Filtering: Approaches

3 main approaches to collaborative filtering

Neighborhood-based
(a.k.a. memory-based)

Hybrid
(memory-model-based)

Latent-factor-based
(a.k.a. model-based)

Neighborhood-based (Memory-based) CF

Compute the relationship between users or items

Neighborhood-based (Memory-based) CF

Compute the relationship between users or items

User-based

Evaluates a user's preference for an item based on ratings of "neighboring" users for that item

Neighborhood-based (Memory-based) CF

Compute the relationship between users or items

User-based

Evaluates a user's preference for an item based on ratings of "neighboring" users for that item

Item-based

Evaluates a user's preference for an item based on ratings of "neighboring" items by the same user

USER-BASED COLLABORATIVE FILTERING

User-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

User-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

From the set of users {u': u' != u} who have already rated i extract a subset of k neighbours of u

User-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

From the set of users {u': u' != u} who have already rated i extract a subset of k neighbours of u

k-neighborhood of u is found on the basis of the similarity between user ratings without explicit user profiles

User-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

From the set of users {u': u' != u} who have already rated i extract a subset of k neighbours of u

k-neighborhood of u is found on the basis of the similarity between user ratings without explicit user profiles

Estimate r(u, i) based on the ratings of users in the k-neighborhood of u

User-based Neighborhood

In theory, rating prediction r(u,i) could be defined on any item i not rated by u

User-based Neighborhood

In theory, rating prediction r(u,i) could be defined on any item i not rated by u

In practice, we are interested only in estimating r(u,i) for those items i which have been rated by the u's k-neighborhood

User-based Neighborhood

In theory, rating prediction r(u,i) could be defined on any item i not rated by u

In practice, we are interested only in estimating r(u,i) for those items i which have been rated by the u's k-neighborhood

Intuitively, if a user v is not in the u's k-neighborhood then very likely u will not be interested in any item that only v has rated

User-based Neighborhood

In theory, rating prediction r(u,i) could be defined on any item i not rated by u

In practice, we are interested only in estimating r(u,i) for those items i which have been rated by the u's k-neighborhood

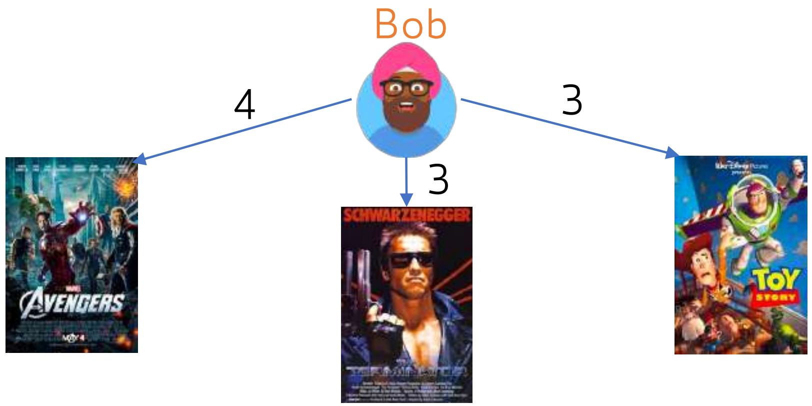
Intuitively, if a user v is not in the u's k-neighborhood then very likely u will not be interested in any item that only v has rated

In other words, the u's k-neighborhood must be computed first to narrow down the set of items which we must predict the rating of

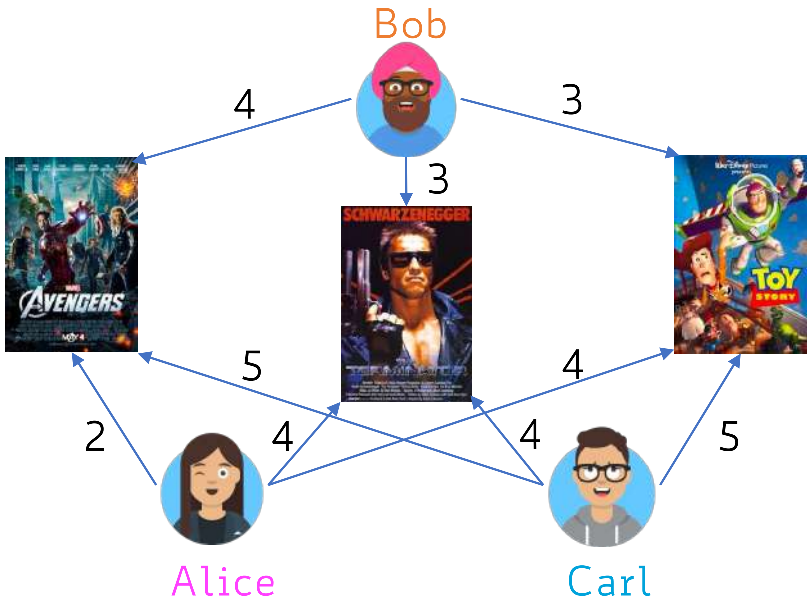
User-based Neighborhood: Example

		MOVIES							
									
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

User-based Neighborhood: Example



User-based Neighborhood: Example



Alice and Carl are the 2-nearest neighbours of Bob
if we look at their rating behaviours

User-based Neighborhood: Example

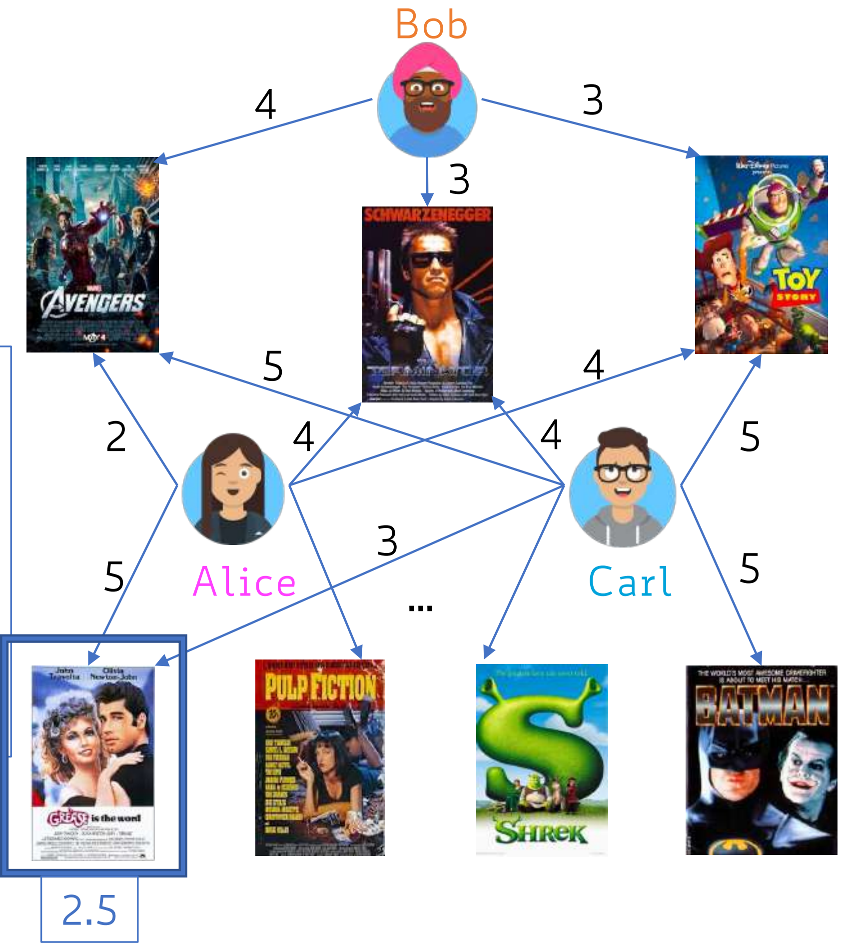
Consider all the movies rated by Alice or Carl which Bob hasn't watched yet

User-based Neighborhood: Example

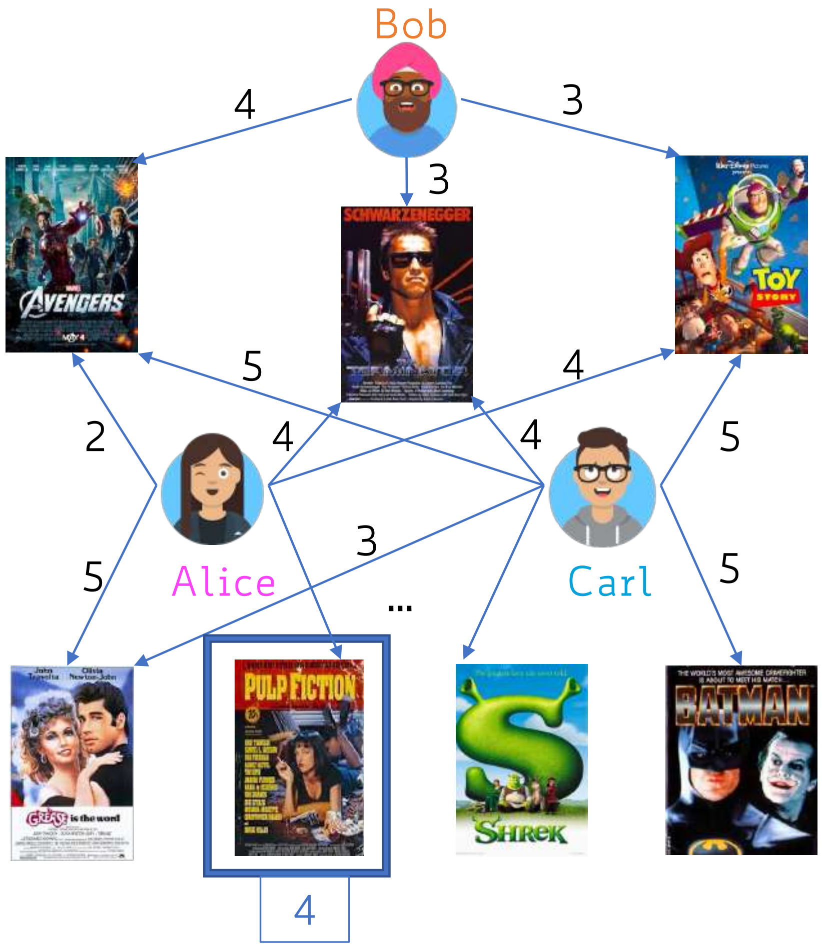
Predict the rating that Bob would give to each of those movies on the basis of Alice's and Carl's ratings

User-based Neighborhood: Example

Predict the rating that Bob would give to each of those movies on the basis of Alice's and Carl's ratings



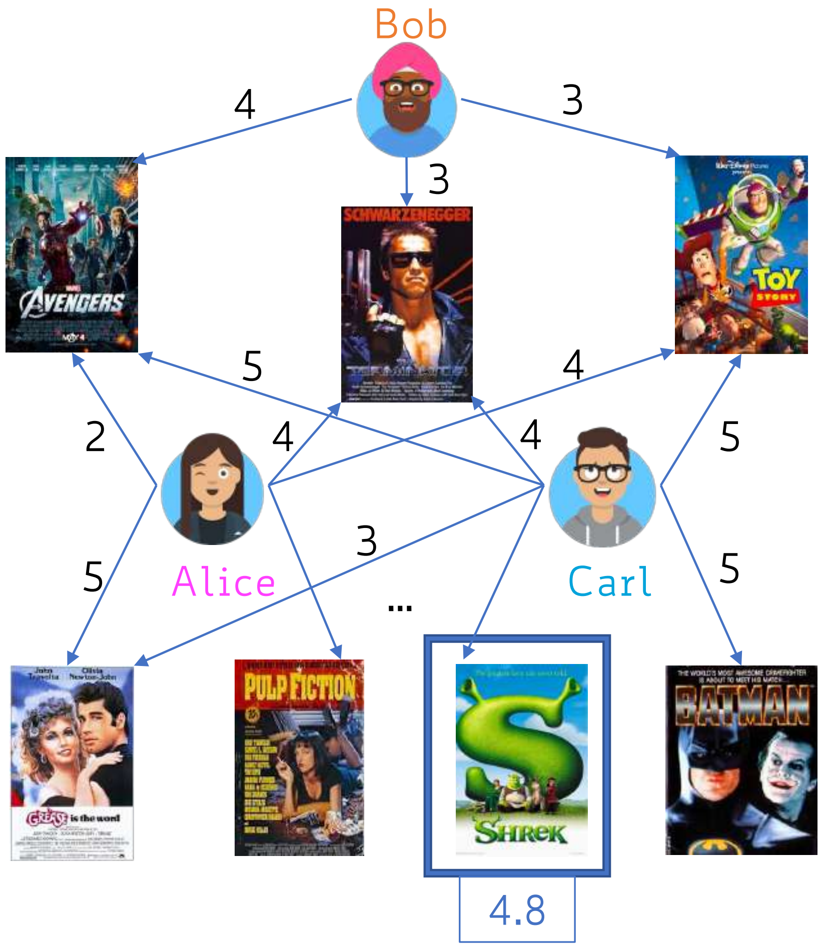
User-based Neighborhood: Example



Predict the rating that Bob would give to each of those movies on the basis of Alice's and Carl's ratings

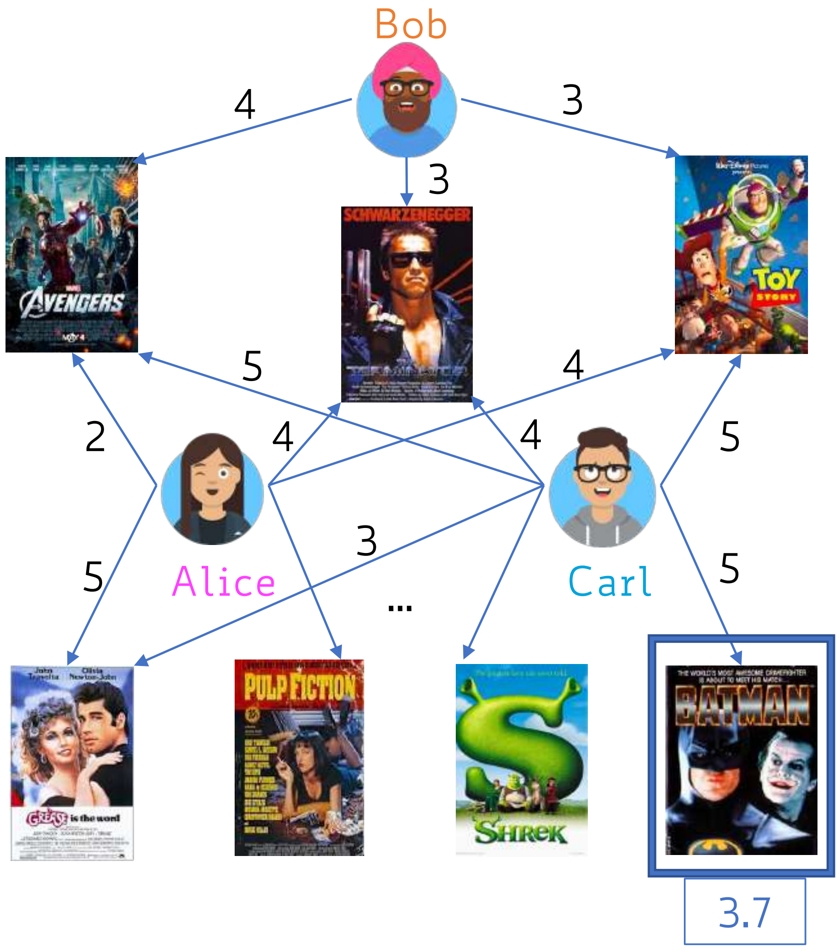
User-based Neighborhood: Example

Predict the rating that Bob would give to each of those movies on the basis of Alice's and Carl's ratings



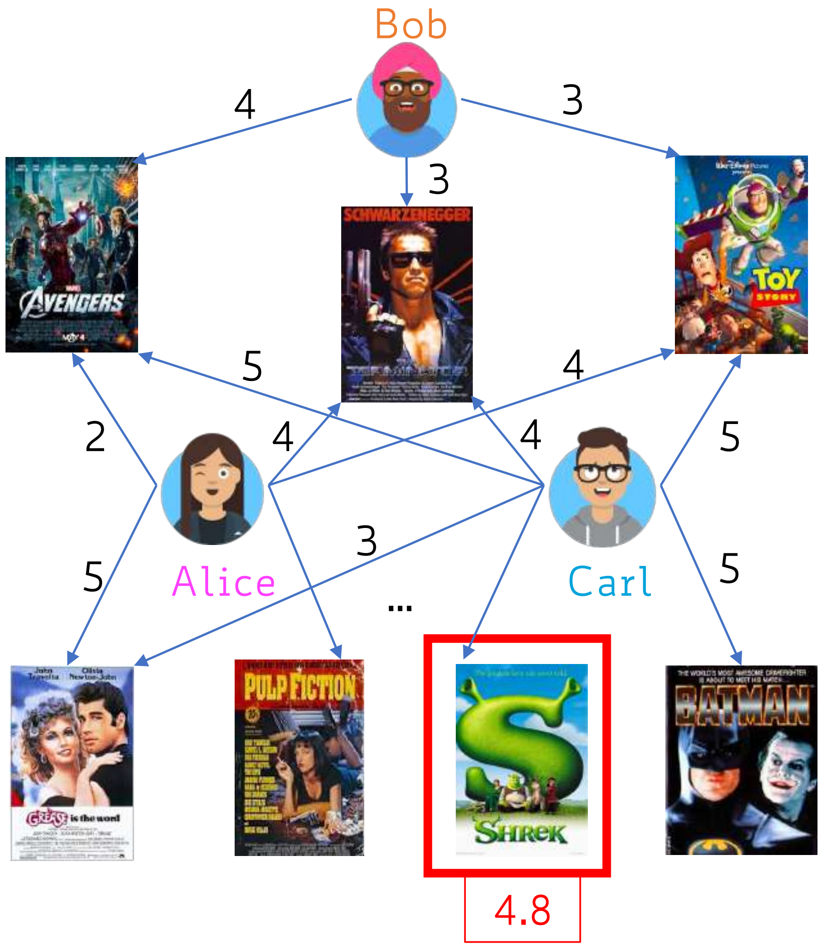
User-based Neighborhood: Example

Predict the rating that Bob would give to each of those movies on the basis of Alice's and Carl's ratings

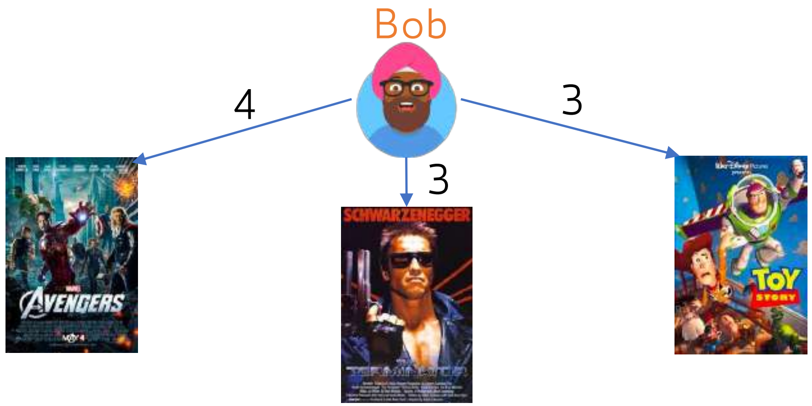


User-based Neighborhood: Example

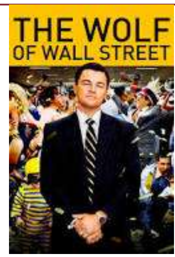
Recommend the highest rated movie(s) to Bob!



User-based Neighborhood: Example



There is no point in predicting the rating of a movie which has only been rated by a user (Zoe) who is not in the Bob's neighborhood



User-to-User Similarity

- The key "trick" to discover the k-neighborhood of a given user u is the ability of finding users u' that are "similar" to u

User-to-User Similarity

- The key "trick" to discover the k-neighborhood of a given user u is the ability of finding users u' that are "similar" to u
- Remember that we are **not** building any content-based user/item profile

User-to-User Similarity

- The key "trick" to discover the k-neighborhood of a given user u is the ability of finding users u' that are "similar" to u
- Remember that we are **not** building any content-based user/item profile
- Intuitively, 2 users u_1 and u_2 are similar to each other if their ratings (of items) are similar

User-to-User Similarity

- The key "trick" to discover the k-neighborhood of a given user u is the ability of finding users u' that are "similar" to u
- Remember that we are **not** building any content-based user/item profile
- Intuitively, 2 users u_1 and u_2 are similar to each other if their ratings (of items) are similar
- Each user represented by her/his rating vector and similarity between them is measured in the item (rating) space

User-to-User Similarity

$\text{sim}(u, v)$ Similarity metric between any pair of users

		MOVIES							
		AVENGERS	BATMAN		PULP FICTION	SHREK	SCHWARZENEGGER	THE WOLF OF WALL STREET	TOY
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

User-to-User Similarity

$\mathrm{sim}(u, v)$ Similarity metric between any pair of users

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

Must capture the intuition: $\underbrace{\mathrm{sim}(\mathrm{Alice}, \mathrm{Carl})} > \underbrace{\mathrm{sim}(\mathrm{Alice}, \mathrm{Bob})}$

User-to-User Similarity

$\mathbf{r}_u$ n-dimensional vector of ratings provided by user u (n = #movies)

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

User-to-User Similarity

$\mathbf{r}_u$ n-dimensional vector of ratings provided by user u (n = #movies)

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

$\mathbf{r}_{\text{Bob}}$

User-to-User Similarity: Jaccard Similarity

$$\mathrm{sim}(u,v) = J(\mathbf{r}_u, \mathbf{r}_v) = \frac{|\mathbf{r}_u \cap \mathbf{r}_v|}{|\mathbf{r}_u \cup \mathbf{r}_v|}$$

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

User-to-User Similarity: Jaccard Similarity

$$\mathrm{sim}(u, v) = J(\mathbf{r}_u, \mathbf{r}_v) = \frac{|\mathbf{r}_u \cap \mathbf{r}_v|}{|\mathbf{r}_u \cup \mathbf{r}_v|}$$

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

$$\mathrm{sim}(\text{Alice}, \text{Bob}) = \frac{|\mathbf{r}_{\text{Alice}} \cap \mathbf{r}_{\text{Bob}}|}{|\mathbf{r}_{\text{Alice}} \cup \mathbf{r}_{\text{Bob}}|}$$

$$= \frac{3}{6} = 0.5$$

User-to-User Similarity: Jaccard Similarity

$$\mathrm{sim}(u,v) = J(\mathbf{r}_u, \mathbf{r}_v) = \frac{|\mathbf{r}_u \cap \mathbf{r}_v|}{|\mathbf{r}_u \cup \mathbf{r}_v|}$$

USERS	MOVIES							
Alice	2		5	4	5	4		4
Bob	4					3		3
Carl	5	5	3	4	5	4		5
…	…	…	…	…	…	…	…	…
Zoe		1	3				5	4

$$\mathrm{sim}(\mathrm{Alice}, \mathrm{Carl}) = \frac{|\mathbf{r}_{\mathrm{Alice}} \cap \mathbf{r}_{\mathrm{Carl}}|}{|\mathbf{r}_{\mathrm{Alice}} \cup \mathbf{r}_{\mathrm{Carl}}|}$$

$$= \frac{6}{7} \approx 0.86$$

User-to-User Similarity: Jaccard Similarity

$$\text{sim}(u, v) = J(\mathbf{r}_u, \mathbf{r}_v) = \frac{|\mathbf{r}_u \cap \mathbf{r}_v|}{|\mathbf{r}_u \cup \mathbf{r}_v|}$$

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

Problem!
Jaccard ignores rating values

User-to-User Similarity: Cosine Similarity

$$\mathrm{sim}(u, v) = \mathrm{cosine}(\mathbf{r}_u, \mathbf{r}_v) = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{||\mathbf{r}_u|| \, ||\mathbf{r}_v||}$$

	MOVIES							
USERS								
Alice	2		5	4	5	4		4
Bob	4					3		3
Carl	5	5	3	4	5	4		5
...	...	...	...	...	...	...	...	...
Zoe		1	3				5	4

$$\mathrm{sim}(\mathrm{Alice}, \mathrm{Bob}) = \frac{\mathbf{r}_{\mathrm{Alice}} \cdot \mathbf{r}_{\mathrm{Bob}}}{||\mathbf{r}_{\mathrm{Alice}}|| \, ||\mathbf{r}_{\mathrm{Bob}}||}$$

$$= \frac{32}{\sqrt{102}\sqrt{44}} \approx 0.48$$

User-to-User Similarity: Cosine Similarity

$$\mathrm{sim}(u, v) = \mathrm{cosine}(\mathbf{r}_u, \mathbf{r}_v) = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{||\mathbf{r}_u||\,||\mathbf{r}_v||}$$

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

$$\mathrm{sim}(\mathrm{Alice}, \mathrm{Carl}) = \frac{\mathbf{r}_{\mathrm{Alice}} \cdot \mathbf{r}_{\mathrm{Carl}}}{||\mathbf{r}_{\mathrm{Alice}}||\,||\mathbf{r}_{\mathrm{Carl}}||}$$

$$= \frac{102}{\sqrt{102}\sqrt{141}} \approx 0.85$$

User-to-User Similarity: Cosine Similarity

$$\text{sim}(u, v) = \text{cosine}(\mathbf{r}_u, \mathbf{r}_v) = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{||\mathbf{r}_u||\,||\mathbf{r}_v||}$$

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

Problem!
Missing rating values are treated as 0s and have a negative effect

User-to-User Similarity: Pearson Correlation

$$\text{sim}(u, v) = \text{Pearson}(\mathbf{r}_u, \mathbf{r}_v) = \frac{(\mathbf{r}_u - \bar{\mathbf{r}}_u) \cdot (\mathbf{r}_v - \bar{\mathbf{r}}_v)}{\sqrt{(\mathbf{r}_u - \bar{\mathbf{r}}_u)^T \cdot (\mathbf{r}_u - \bar{\mathbf{r}}_u)} \times \sqrt{(\mathbf{r}_v - \bar{\mathbf{r}}_v)^T \cdot (\mathbf{r}_v - \bar{\mathbf{r}}_v)}}$$

		MOVIES							
USERS	Alice	-2		1	0	1	0		0
	Bob	2/3					-1/3		-1/3
	Carl	4/7	4/7	-10/7	-3/7	4/7	-3/7		4/7
	…	…	…	…	…	…	…	…	…
	Zoe		-9/4	-1/4				7/4	-1/4

Solution: Normalize ratings by subtracting the mean rating

Now 0 means neutral, and if we treat missing ratings as 0, it doesn't mean it's negative

User-to-User Similarity: Pearson Correlation

$\mathbf{r}'_u = \mathbf{r}_u - \bar{\mathbf{r}}_u$ mean-scaled rating vector of u

$\mathbf{r}'_v = \mathbf{r}_v - \bar{\mathbf{r}}_v$ mean-scaled rating vector of v

User-to-User Similarity: Pearson Correlation

$\mathbf{r}'_u = \mathbf{r}_u - \bar{\mathbf{r}}_u$ mean-scaled rating vector of u

$\mathbf{r}'_v = \mathbf{r}_v - \bar{\mathbf{r}}_v$ mean-scaled rating vector of v

$$\text{cosine}(\mathbf{r}'_u, \mathbf{r}'_v) = \frac{\mathbf{r}'_u \cdot \mathbf{r}'_v}{||\mathbf{r}'_u|| \, ||\mathbf{r}'_v||} = \frac{(\mathbf{r}_u - \bar{\mathbf{r}}_u) \cdot (\mathbf{r}_v - \bar{\mathbf{r}}_v)}{||\mathbf{r}_u - \bar{\mathbf{r}}_u|| \, ||\mathbf{r}_v - \bar{\mathbf{r}}_v||} =$$

User-to-User Similarity: Pearson Correlation

$\mathbf{r}'_u = \mathbf{r}_u - \bar{\mathbf{r}}_u$ mean-scaled rating vector of u

$\mathbf{r}'_v = \mathbf{r}_v - \bar{\mathbf{r}}_v$ mean-scaled rating vector of v

$$\text{cosine}(\mathbf{r}'_u, \mathbf{r}'_v) = \frac{\mathbf{r}'_u \cdot \mathbf{r}'_v}{||\mathbf{r}'_u||||\mathbf{r}'_v||} = \frac{(\mathbf{r}_u - \bar{\mathbf{r}}_u) \cdot (\mathbf{r}_v - \bar{\mathbf{r}}_v)}{||\mathbf{r}_u - \bar{\mathbf{r}}_u||||\mathbf{r}_v - \bar{\mathbf{r}}_v||} =$$

$$= \frac{(\mathbf{r}_u - \bar{\mathbf{r}}_u) \cdot (\mathbf{r}_v - \bar{\mathbf{r}}_v)}{\sqrt{(\mathbf{r}_u - \bar{\mathbf{r}}_u)^T \cdot (\mathbf{r}_u - \bar{\mathbf{r}}_u)} \times \sqrt{(\mathbf{r}_v - \bar{\mathbf{r}}_v)^T \cdot (\mathbf{r}_v - \bar{\mathbf{r}}_v)}} = \text{Pearson}(\mathbf{r}_u, \mathbf{r}_v)$$

User-based Neighborhood: Predictions

$\mathbf{r}_u$ Vector of ratings provided by user u

User-based Neighborhood: Predictions

$\mathbf{r}_u$ Vector of ratings provided by user u

$$\mathcal{U}^k = \text{argmax}_{\mathcal{U}' \subseteq \mathcal{U} \setminus u, |\mathcal{U}'| = k} \sum_{u' \in \mathcal{U}'} \text{sim}(u, u')$$

Top-k most "similar" users to u

u's k-neighborhood

User-based Neighborhood: Predictions

$\mathbf{r}_u$ Vector of ratings provided by user u

$$\mathcal{U}^k = \text{argmax}_{\mathcal{U}' \subseteq \mathcal{U} \setminus u, |\mathcal{U}'|=k} \sum_{u' \in \mathcal{U}'} \text{sim}(u, u')$$

Top-k most "similar" users to u

u's k-neighborhood

Set of items rated by u's neighbors

$$\mathcal{I}^k = \{i \in \mathcal{I} : \mathbf{r}_{u',i} = \downarrow \wedge u' \in \mathcal{U}^k\}$$

User-based Neighborhood: Predictions

$\mathbf{r}_u$ Vector of ratings provided by user u

$$\mathcal{U}^k = \mathrm{argmax}_{\mathcal{U}' \subseteq \mathcal{U} \setminus u, |\mathcal{U}'|=k} \sum_{u' \in \mathcal{U}'} \mathrm{sim}(u, u')$$

Top-k most "similar" users to u

u's k-neighborhood

Set of items rated by u's neighbors

$$\mathcal{I}^k = \{i \in \mathcal{I} : \mathbf{r}_{u',i} =\downarrow \wedge \, u' \in \mathcal{U}^k\}$$

Predicted rating given by user u to item i

$$\mathbf{r}_u[i] = r(u, i) = r_{u,i}$$

User-based Neighborhood: Predictions

2 possible ways of aggregating neighbors ratings

User-based Neighborhood: Predictions

$\boxed{\text{2 possible ways of aggregating neighbors ratings}}$

$$\forall i \in \mathcal{I}^k$$

User-based Neighborhood: Predictions

2 possible ways of aggregating neighbors ratings

$$\forall i \in \mathcal{I}^k$$

$$r_{u,i} = \frac{1}{k} \sum_{u' \in \mathcal{U}^k} r_{u',i}$$

plain average

User-based Neighborhood: Predictions

2 possible ways of aggregating neighbors ratings

$$\forall i \in \mathcal{I}^k$$

$$r_{u,i} = \frac{1}{k} \sum_{u' \in \mathcal{U}^k} r_{u',i}$$

plain average

$$r_{u,i} = \frac{1}{k} \sum_{u' \in \mathcal{U}^k} \mathrm{sim}(u, u') \cdot r_{u',i}$$

weighted average

User-based CF: Drawbacks

$\boxed{\text{3 main issues with user-based CF}}$

User-based CF: Drawbacks

3 main issues with user-based CF

Sparsity

systems performed poorly when they had many items but comparatively few ratings

User-based CF: Drawbacks

3 main issues with user-based CF

Sparsity
systems performed poorly when they had many items but comparatively few ratings

Efficiency
computing similarities between all pairs of users is expensive

User-based CF: Drawbacks

3 main issues with user-based CF

Sparsity
systems performed poorly when they had many items but comparatively few ratings

Efficiency
computing similarities between all pairs of users is expensive

Aging
user profiles changed quickly and the entire system model had to be recomputed

ITEM-BASED COLLABORATIVE FILTERING

Item-based CF

- Introduced by Amazon to overcome the 3 issues of user-based CF

Item-based CF

- Introduced by Amazon to overcome the 3 issues of user-based CF
- Since systems have typically more users than items, each item has more ratings than each user

Item-based CF

- Introduced by [Amazon] to overcome the 3 issues of user-based CF
- Since systems have typically more users than items, each item has more ratings than each user
- As such, an item's rating average is **more stable** over time

Item-based CF

- Introduced by [Amazon]() to overcome the 3 issues of user-based CF
- Since systems have typically more users than items, each item has more ratings than each user
- As such, an item's rating average is **more stable** over time
- The model doesn't suffer from aging and therefore it does not need to be recomputed frequently

Item-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

Item-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

From the set of items already rated by u (I_u)
extract a subset of k neighbours of i

Item-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

From the set of items already rated by u (I_u)
extract a subset of k neighbours of i

k-neighborhood is found on the basis of the similarity between items without explicit item's content or metadata

Item-based Neighborhood

Given a user u and an item i not rated by u, we want to estimate r(u, i)

From the set of items already rated by u (I_u)
extract a subset of k neighbours of i

k-neighborhood is found on the basis of the similarity
between items without explicit item's content or metadata

Estimate r(u, i) based on the ratings of items in the
k-neighborhood of i

Item-to-Item Similarity

- The key "trick" to discover the k-neighborhood of a given item i is the ability of finding items that are "similar" to i

Item-to-Item Similarity

- The key "trick" to discover the *k*-neighborhood of a given item *i* is the ability of finding items that are "similar" to *i*
- Remember that we are not building any content-based user/item profile

Item-to-Item Similarity

- The key "trick" to discover the *k*-neighborhood of a given item *i* is the ability of finding items that are "similar" to *i*
- Remember that we are not building any content-based user/item profile
- Intuitively, 2 items i_1 and i_2 are similar to each other if users who rated them are similar

Item-to-Item Similarity

- The key "trick" to discover the *k*-neighborhood of a given item i is the ability of finding items that are "similar" to i
- Remember that we are not building any content-based user/item profile
- Intuitively, 2 items i_1 and i_2 are similar to each other if users who rated them are similar
- Each item represented by the user ratings vector and similarity between them is measured in the user (rating) space

Item-to-Item Similarity

$\mathbf{r}_i$ m-dimensional vector of ratings provided for item i (m = #users)

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

Item-to-Item Similarity

$\mathbf{r}_i$ m-dimensional vector of ratings provided for item i (m = #users)

		MOVIES							
		AVENGERS	BATMAN	Grease	PULP FICTION	SHREK	SCHWARZENEGGER	THE WOLF OF WALL STREET	TOY
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

$\mathbf{r}_{\text{Shrek}}$

Item-based Neighborhood: Example

Let's consider again Bob!

		MOVIES							
									
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

Item-based Neighborhood: Example

Suppose we want to predict the rating Bob would give to Shrek

USERS	MOVIES							
Alice	2		5	4	5	4		4
Bob	4				?	3		3
Carl	5	5	3	4	5	4		5
...	...	...	...	...	...	...	...	...
Zoe		1	3				5	4

Item-based Neighborhood: Example

We first extract the subset of k most similar items to Shrek which have been rated by Bob

		MOVIES							
		AVENGERS	BATMAN	Grease	PULP FICTION	SHREK	SCHWARZENEGGER	THE WOLF OF WALL STREET	TOY
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

$\mathbf{r}_{\text{Shrek}}$

Item-based Neighborhood: Example

Suppose those are: The Avengers and The Terminator

		MOVIES							
USERS	Alice	2		5	4	5	4		4
	Bob	4					3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

For example, item similarity is measured using Pearson's correlation

Item-based Neighborhood: Example

The predicted rating is computed as an aggregating function of the ratings that Bob gave to the k most similar movies to Shrek

		MOVIES							
									
USERS	Alice	2		5	4	5	4		4
	Bob	4				?	3		3
	Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	Zoe		1	3				5	4

Item-based Neighborhood: Predictions

$\mathbf{r}_i$ Vector of ratings given to item i

Item-based Neighborhood: Predictions

$\mathbf{r}_i$ Vector of ratings given to item i

$\mathcal{I}_u = \{i \in \mathcal{I} : r_{u,i} = \downarrow\}$ Set of items rated by u

Item-based Neighborhood: Predictions

$\mathbf{r}_i$ Vector of ratings given to item i

$\mathcal{I}_u = \{i \in \mathcal{I} : r_{u,i} = \downarrow\}$ Set of items rated by u

$$\mathcal{I}_u^k = \text{argmax}_{\mathcal{I}'_u \subseteq \mathcal{I}_u, |\mathcal{I}'_u| = k} \sum_{i' \in \mathcal{I}'_u} \text{sim}(i, i')$$

Top-k most "similar" items to i among those rated by u

i's k-neighborhood

Item-based Neighborhood: Predictions

$\mathbf{r}_i$ Vector of ratings given to item i

$$\mathcal{I}_u = \{i \in \mathcal{I} : r_{u,i} = \downarrow\}$$

Set of items rated by u

$$\mathcal{I}_u^k = \mathrm{argmax}_{\mathcal{I}'_u \subseteq \mathcal{I}_u, |\mathcal{I}'_u| = k} \sum_{i' \in \mathcal{I}'_u} \mathrm{sim}(i, i')$$

Top-k most "similar" items to i among those rated by u

i's k-neighborhood

Predicted rating given by user u to item i

$$\mathbf{r}_u[i] = r(u, i) = r_{u,i}$$

Item-based Neighborhood: Predictions

2 possible ways of aggregating neighbors ratings

Item-based Neighborhood: Predictions

2 possible ways of aggregating neighbors ratings

$$\forall i' \in \mathcal{I}_u^k$$

$$\boxed{r_{u,i} = \frac{1}{k} \sum_{i' \in \mathcal{I}_u^k} r_{u,i'}}$$

plain average

Item-based Neighborhood: Predictions

2 possible ways of aggregating neighbors ratings

$$\forall i' \in \mathcal{I}_u^k$$

$$r_{u,i} = \frac{1}{k} \sum_{i' \in \mathcal{I}_u^k} r_{u,i'}$$

plain average

$$r_{u,i} = \frac{1}{k} \sum_{i' \in \mathcal{I}_u^k} \mathrm{sim}(i, i') \cdot r_{u,i'}$$

weighted average

Item-to-Item Collaborative Filtering

- Item similarity can be computed from rating vectors (in user space)

Item-to-Item Collaborative Filtering

- Item similarity can be computed from rating vectors (in user space)
- Analogous to user similarity of rating vectors (in item space):
  - Jaccard index
  - Cosine similarity (normalized = Pearson's correlation)

Item-to-Item Collaborative Filtering

- Item similarity can be computed from rating vectors (in user space)
- Analogous to user similarity of rating vectors (in item space):
  - Jaccard index
  - Cosine similarity (normalized = Pearson's correlation)
- Rating prediction using the same methods proposed for user-based CF
  - Plain average of ratings
  - Weighted average of ratings (taking item similarity into account)

Item-to-Item Collaborative Filtering

In general, item-based works better than user-based CF

Memory-based CF: Implementation

- The most expensive step is finding the k most similar users (or items)

Memory-based CF: Implementation

- The most expensive step is finding the k most similar users (or items)
- This computation is too expensive to do online (for every user/item)

Memory-based CF: Implementation

- The most expensive step is finding the k most similar users (or items)
- This computation is too expensive to do online (for every user/item)
- Finding the k most similar users/items should be pre-computed (offline)

Memory-based CF: Implementation

- The most expensive step is finding the k most similar users (or items)
- This computation is too expensive to do online (for every user/item)
- Finding the k most similar users/items should be pre-computed (offline)
- k-nearest neighbors search in high dimensions (i.e., quickly find the set of k nearest data points)

Memory-based CF: Implementation

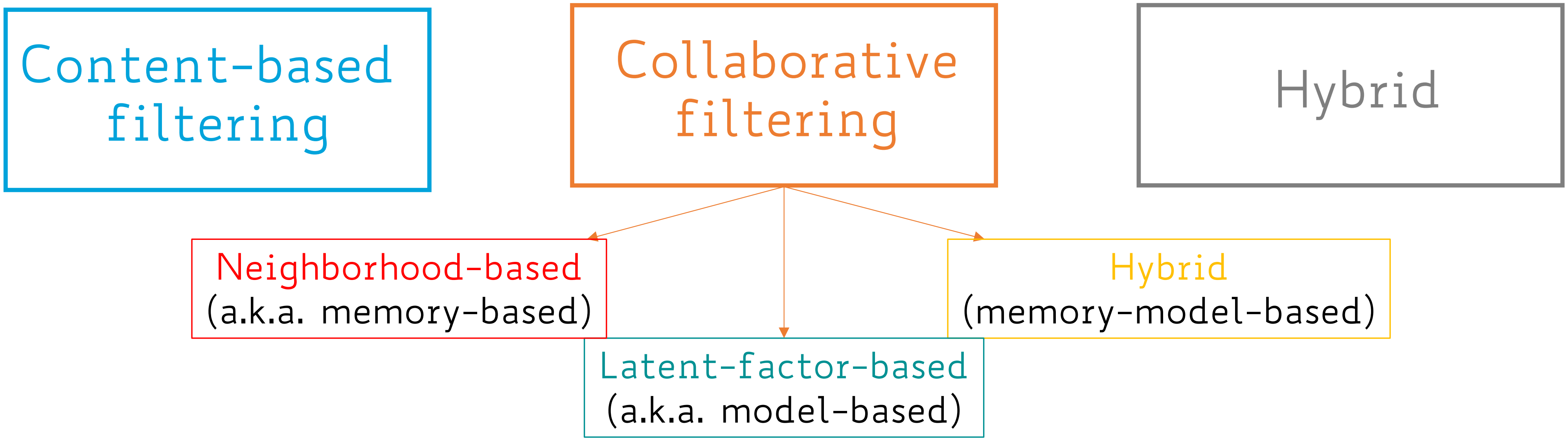
The curse of dimensionality (again!)

Memory-based CF: Implementation

Locality-Sensitive Hashing (LSH) approximation

Recommendation Strategies

3 approaches to recommender systems



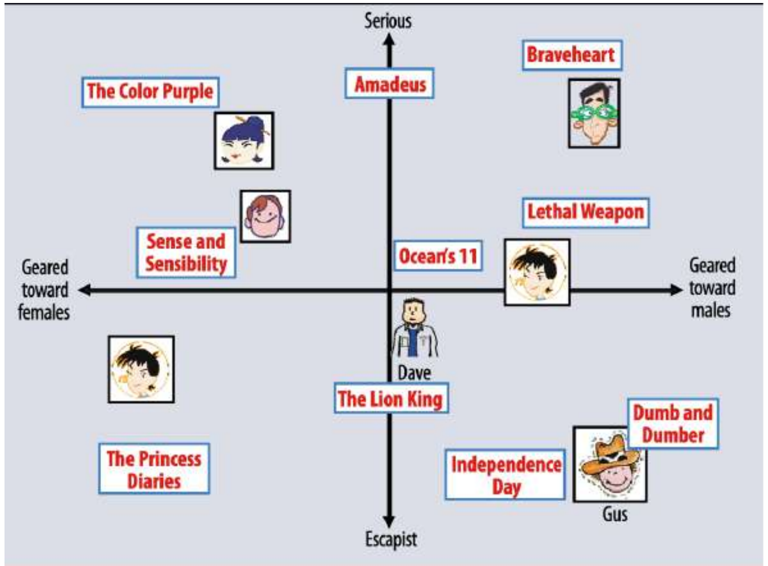
LATENT FACTOR MODELS

Latent Factor (Model-based) CF

Tries to predict ratings by representing both items and users with a number of hidden factors inferred from observed ratings

Latent Factor (Model-based) CF

Tries to predict ratings by representing both items and users with a number of hidden factors inferred from observed ratings

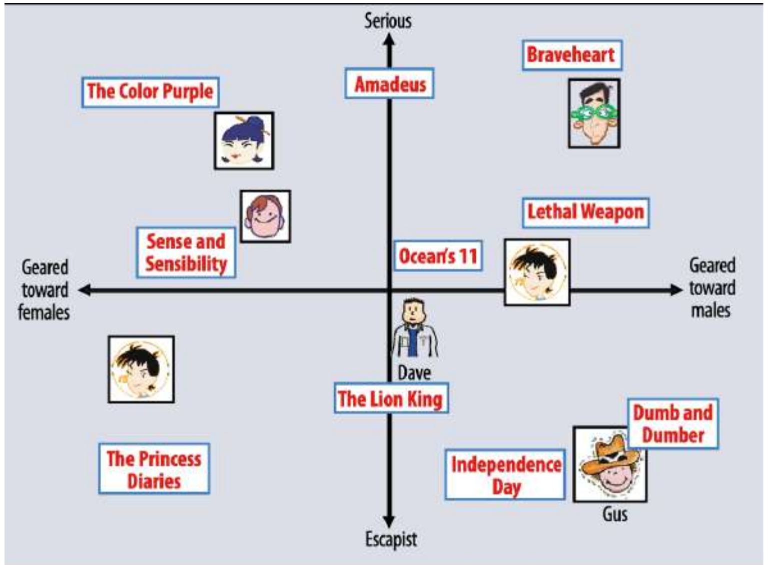


Example: 2 hidden factors

- Dim. 1: Male vs. Female
- Dim. 2: Serious vs. Escapist

Latent Factor (Model-based) CF

Tries to predict ratings by representing both items and users with a number of hidden factors inferred from observed ratings



Example: 2 hidden factors

- Dim. 1: Male vs. Female
- Dim. 2: Serious vs. Escapist

A user's predicted rating for an item (movie) would equal the dot product of the movie and user vectors on the plot

Matrix Factorization

- Some of the most successful realizations of latent factor models are based on **matrix factorization** (MF)

Matrix Factorization

- Some of the most successful realizations of latent factor models are based on **matrix factorization** (MF)
- The original idea behind MF is to represent users and items in a lower dimensional latent space (i.e., as **vectors of latent factors**)

Matrix Factorization

- Some of the most successful realizations of latent factor models are based on **matrix factorization** (MF)
- The original idea behind MF is to represent users and items in a lower dimensional latent space (i.e., as **vectors of latent factors**)
- Such vectors are inferred (i.e., learned) from observed item ratings

Matrix Factorization

- Some of the most successful realizations of latent factor models are based on **matrix factorization** (MF)
- The original idea behind MF is to represent users and items in a lower dimensional latent space (i.e., as **vectors of latent factors**)
- Such vectors are inferred (i.e., learned) from observed item ratings
- High correspondence between item and user factors leads to a recommendation

Matrix Factorization Framework

- Map both items and users to a **joint latent factor** d-dimensional space

Matrix Factorization Framework

- Map both items and users to a **joint latent factor** d-dimensional space
- User-Item interactions are modeled as **inner products** in that space

Matrix Factorization Framework

- Map both items and users to a **joint latent factor** *d*-dimensional space
- User-Item interactions are modeled as **inner products** in that space

$\mathbf{x}_u \in \mathbb{R}^d$ d-dimensional vector representing user u

Each $x_u[k]$ measures the extent of interest user u has in items exhibiting the k-th factor

Matrix Factorization Framework

- Map both items and users to a **joint latent factor** *d*-dimensional space
- User-Item interactions are modeled as **inner products** in that space

$\mathbf{x}_u \in \mathbb{R}^d$ d-dimensional vector representing user u

$\mathbf{w}_i \in \mathbb{R}^d$ d-dimensional vector representing item i

Each $x_u[k]$ measures the extent of interest user u has in items exhibiting the k-th factor

Each $w_i[k]$ measures the extent to which the item i has the k-th factor

What Are Those Factors?

- Essentially, d hidden features for describing both users and items

What Are Those Factors?

- Essentially, d hidden features for describing both users and items
- In the user-movie example, a feature f may refer to:
  - how much each a user likes Disney movies (in the case of user vectors)
  - how close a movie is to a Disney movie (in the case of item, i.e., movie, vectors)

What Are Those Factors?

- Essentially, d hidden features for describing both users and items
- In the user-movie example, a feature f may refer to:
  - how much each a user likes Disney movies (in the case of user vectors)
  - how close a movie is to a Disney movie (in the case of item, i.e., movie, vectors)
- We do not know what these features are nor do we have to determine them beforehand!

What Are Those Factors?

- Essentially, d hidden features for describing both users and items
- In the user-movie example, a feature f may refer to:
  - how much each a user likes Disney movies (in the case of user vectors)
  - how close a movie is to a Disney movie (in the case of item, i.e., movie, vectors)
- We do not know what these features are nor do we have to determine them beforehand!
- That is why these features are often refer to as **latent features**

Matrix Factorization Framework

$r(u, i) = r_{u,i}$ rating of user u for the item i

Matrix Factorization Framework

$r(u, i) = r_{u,i}$ rating of user u for the item i

$\hat{r}_{u,i} = \mathbf{x}_u^T \cdot \mathbf{w}_i = \sum_{j=1}^{d} x_{u,j} w_{j,i}$ estimated (i.e., predicted) rating of user u for the item i

Matrix Factorization Framework

$r(u, i) = r_{u,i}$ rating of user u for the item i

$\hat{r}_{u,i} = \mathbf{x}_u^T \cdot \mathbf{w}_i = \sum_{j=1}^{d} x_{u,j} w_{j,i}$ estimated (i.e., predicted) rating of user u for the item i

The major challenge is computing the mapping of each item and user to latent factor vectors x_u and w_i

Matrix Factorization Framework

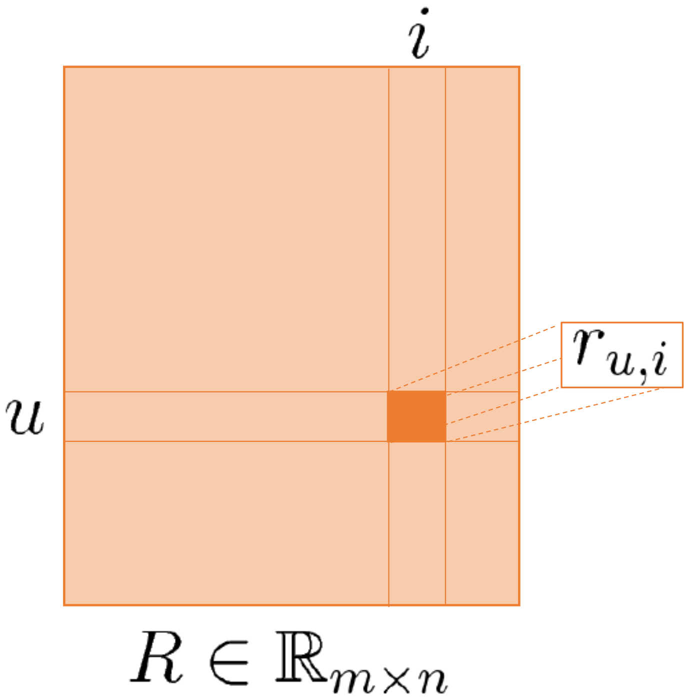
$r(u, i) = r_{u,i}$ rating of user u for the item i

$\hat{r}_{u,i} = \mathbf{x}_u^T \cdot \mathbf{w}_i = \sum_{j=1}^{d} x_{u,j} w_{j,i}$ estimated (i.e., predicted) rating of user u for the item i

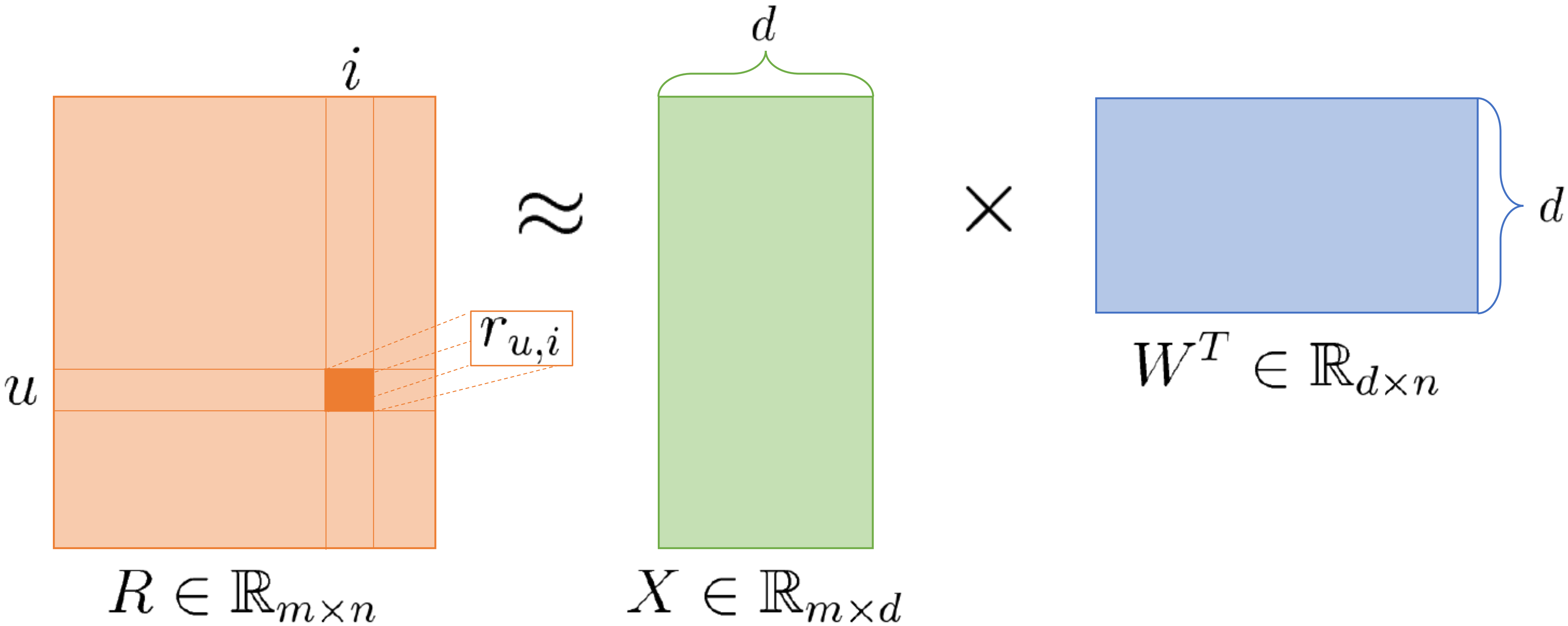
The major challenge is computing the mapping of each item and user to latent factor vectors x_u and w_i

Recommendations for a user are generated by computing the estimated ratings for unseen items, and by taking the top-k highest rated ones

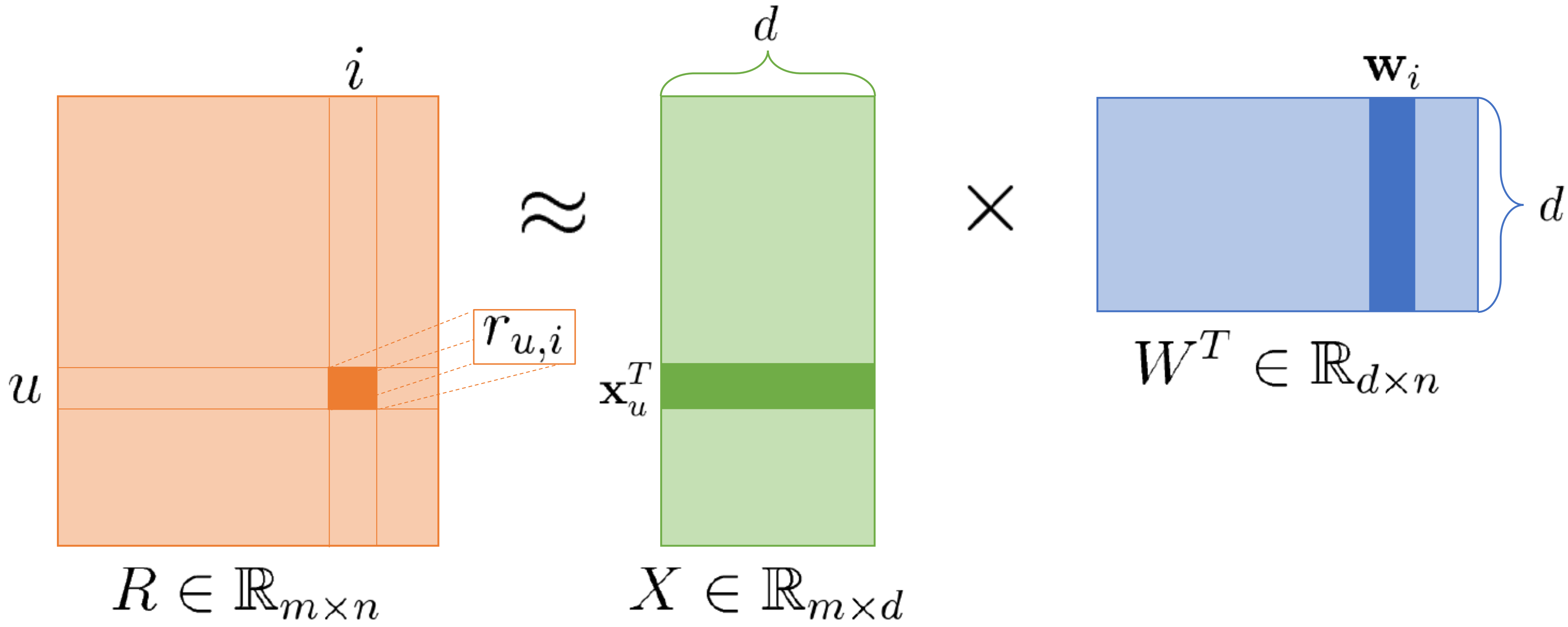
Matrix Factorization Framework



Matrix Factorization Framework



Matrix Factorization Framework



Approximate the user-item rating matrix R with the product of X x W^T

How Do We Learn X and W?

Assuming we have access to a dataset of observed ratings

How Do We Learn X and W?

Assuming we have access to a dataset of observed ratings

The matrix R is partially known and filled with those observations

How Do We Learn X and W?

Assuming we have access to a dataset of observed ratings

The matrix R is partially known and filled with those observations

To actually learn the latent factor representations x_u and w_i we minimize the following loss function

$$L(X, W) = \underbrace{\sum_{(u,i) \in \mathcal{D}}}_{\text{Training set of observed ratings}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i \right)^2 + \lambda \left(\sum_{u \in \mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i \in \mathcal{D}} ||\mathbf{w}_i||^2 \right)$$

How Do We Learn X and W?

Assuming we have access to a dataset of observed ratings

The matrix R is partially known and filled with those observations

To actually learn the latent factor representations x_u and w_i we minimize the following loss function

$$L(X,W) = \sum_{(u,i)\in\mathcal{D}} \underbrace{\boxed{\left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i\right)^2}}_{\text{squared error term}} + \lambda\left(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2\right)$$

Training set of observed ratings

How Do We Learn X and W?

Assuming we have access to a dataset of observed ratings

The matrix R is partially known and filled with those observations

To actually learn the latent factor representations x_u and w_i we minimize the following loss function

$$L(X, W) = \sum_{(u,i) \in \mathcal{D}} \underbrace{\left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i \right)^2}_{\text{squared error term}} + \underbrace{\lambda \left(\sum_{u \in \mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i \in \mathcal{D}} ||\mathbf{w}_i||^2 \right)}_{\text{regularization term}}$$

Training set of observed ratings

Matrix Factorization: Optimization

$$X^*, W^* = \text{argmin}_{X,W} L(X, W)$$

Matrix Factorization: Optimization

$$X^*, W^* = \operatorname{argmin}_{X,W} L(X, W)$$

$$X^*, W^* = \operatorname{argmin}_{X,W} \sum_{(u,i)\in\mathcal{D}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i \right)^2 + \lambda \left(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2 \right)$$

Matrix Factorization: Optimization

$$X^*, W^* = \text{argmin}_{X,W} L(X, W)$$

$$X^*, W^* = \text{argmin}_{X,W} \sum_{(u,i)\in\mathcal{D}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i\right)^2 + \lambda\Big(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2\Big)$$

$$X^*, W^* = \text{argmin}_{X,W} \left\{\frac{1}{2} \sum_{(u,i)\in\mathcal{D}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i\right)^2 + \lambda\Big(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2\Big)\right\}$$

Matrix Factorization: Optimization

$$X^*, W^* = \text{argmin}_{X,W} L(X, W)$$

$$X^*, W^* = \text{argmin}_{X,W} \sum_{(u,i)\in\mathcal{D}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i\right)^2 + \lambda\left(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2\right)$$

$$X^*, W^* = \text{argmin}_{X,W} \left\{\boxed{\frac{1}{2}} \sum_{(u,i)\in\mathcal{D}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i\right)^2 + \lambda\left(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2\right)\right\}$$

Mathematically convenient

Matrix Factorization: Optimization

$$X^*, W^* = \text{argmin}_{X,W} L(X, W)$$

$$X^*, W^* = \text{argmin}_{X,W} \sum_{(u,i)\in\mathcal{D}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i\right)^2 + \lambda\Big(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2\Big)$$

$$X^*, W^* = \text{argmin}_{X,W} \left\{ \frac{1}{2} \sum_{(u,i)\in\mathcal{D}} \left(r_{u,i} - \mathbf{x}_u^T \cdot \mathbf{w}_i\right)^2 + \lambda\Big(\sum_{u\in\mathcal{D}} ||\mathbf{x}_u||^2 + \sum_{i\in\mathcal{D}} ||\mathbf{w}_i||^2\Big) \right\}$$

Still, how do we solve this?

Take-Home Message of Today

- Collaborative-filtering may require computing the all-pair user/item similarity

Take-Home Message of Today

- Collaborative-filtering may require computing the all-pair user/item similarity
- In large scale systems, this must be pre-computed offline

Take-Home Message of Today

- Collaborative-filtering may require computing the all-pair user/item similarity
- In large scale systems, this must be pre-computed offline
- At inference time, make use of ad hoc data structures (e.g., k-d trees) to efficiently compute the set of (approximated) nearest neighbors for a query user/item

Take-Home Message of Today

- Collaborative-filtering may require computing the all-pair user/item similarity
- In large scale systems, this must be pre-computed offline
- At inference time, make use of ad hoc data structures (e.g., k-d trees) to efficiently compute the set of (approximated) nearest neighbors for a query user/item
- Latent Factor Models overcome this need