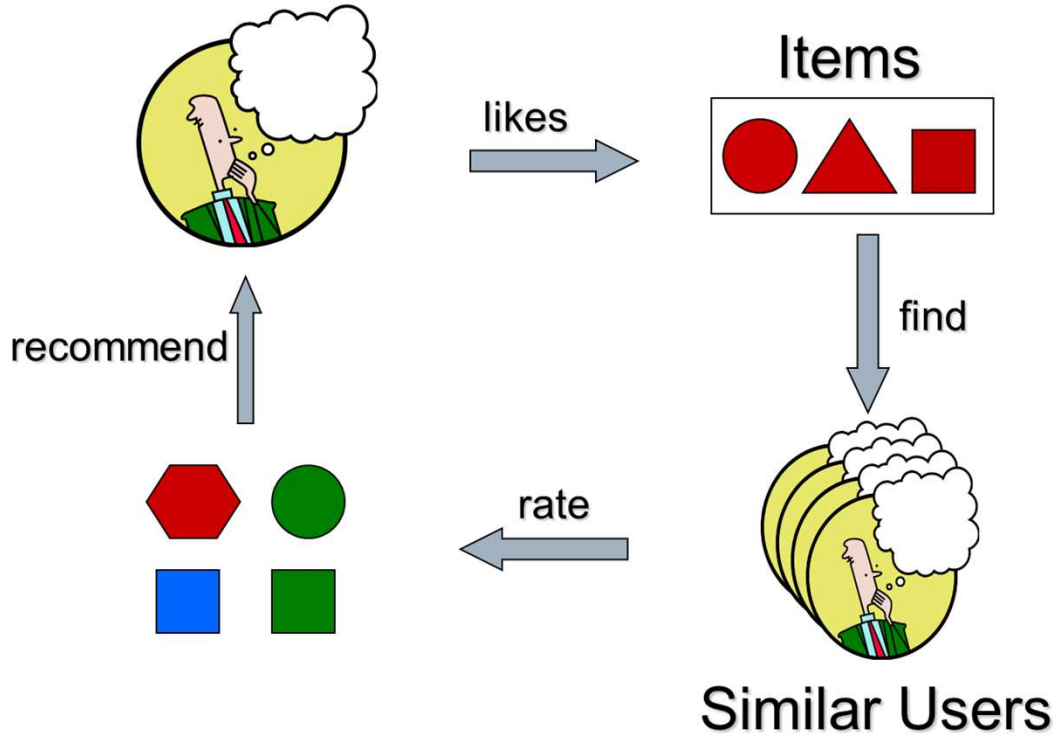


# Parte III

## Collaborative Filtering

# Schema del metodo



Consideriamo l'utente  $x$

Troviamo  $N$  altri utenti i cui rating sono "**simili**" ai rating di  $x$

Stimiamo i rating di  $x$  sulla base degli  $N$  rating degli utenti

# Come misuriamo la similarità

$$\begin{aligned} r_x &= [* , \_ , \_ , * , ***] \\ r_y &= [* , \_ , ** , ** , \_] \end{aligned}$$

- **Jaccard similarity:**

$$\text{sim}(x, y) = |r_x \cap r_y| / |r_x \cup r_y|$$

- **Cosine similarity:**

$$\text{sim}(x, y) = \cos(r_x, r_y)$$

- **Pearson correlation coefficient:**

$$\text{sim}(x, y) = \frac{\sum_{s \in S_{xy}} (r_x[s] - \bar{r}_x)(r_y[s] - \bar{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_x[s] - \bar{r}_x)^2 (r_y[s] - \bar{r}_y)^2}}$$

# Predizioni

## Dalla similarità alla raccomandazione:

- Sia  $\mathbf{r}_x$  il vettore di rating dell'utente  $x$
- Sia  $N$  l'insieme dei  $k$  utenti più simili a  $x$  che hanno valutato l'oggetto  $i$

- **Predizione dell'oggetto  $i$  per l'utente  $x$ :**

- $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$

- $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

$$s_{xy} = \text{sim}(x, y)$$

- Altre opzioni?

- Molte..

# Oggetto-Oggetto Collaborative Filtering

- **Vista: oggetto-oggetto**

- Troviamo gli oggetti simili a  $i$
- Stimiamo il rating per  $i$  in base ai rating degli oggetti simili

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

$s_{ij}$ ... similarità degli oggetti  $i$  e  $j$

$r_{xj}$ ...rating dell'utente  $x$  sull'oggetto  $j$

$N(i;x)$ ... set di oggetti simili a  $i$  valutati da  $x$

# Oggetto-Oggetto CF ( $|N|=2$ )

utenti

film

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



- Rating  
sconosciuto



- rating tra 1 e 5

# Oggetto-Oggetto CF ( $|N|=2$ )

utenti

film

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



- Stimare il rating del film 1 per l'utente 5

# Oggetto-Oggetto CF ( $|N|=2$ )

		utenti												sim(1,m)
		1	2	3	4	5	6	7	8	9	10	11	12	
film	1	1		3		?	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	<u>6</u>	1		3		3			2			4		<u>0.59</u>

## Neighbor selection:

Identificare film simili al film 1,  
valutati dall'utente 5

Usiamo la Pearson correlation come :

1) Sottraiamo il rating medio  $m_i$  da ogni film  $i$

$$m_1 = (1+3+5+5+4)/5 = 3.6$$

row 1: [-2.6, 0, -0.6, 0, 0, 1.4, 0, 0, 1.4, 0, 0.4, 0]

2) Calcoliamo la cosine similarity



# Oggetto-Oggetto CF ( $|N|=2$ )

		utenti												sim(1,m)
		1	2	3	4	5	6	7	8	9	10	11	12	
film	1	1		3		?	5			5		4		1.00
	2			5	4			4			2	1	3	-0.18
	<u>3</u>	2	4		1	2		3		4	3	5		<u>0.41</u>
	4		2	4		5			4			2		-0.10
	5			4	3	4	2					2	5	-0.31
	<u>6</u>	1		3		3			2			4		<u>0.59</u>

Pesi similarità:

$$s_{1,3}=0.41, s_{1,6}=0.59$$

## Oggetto-Oggetto CF ( $|N|=2$ )

		utenti											
		1	2	3	4	5	6	7	8	9	10	11	12
film	1	1		3		2.6	5			5		4	
	2			5	4			4			2	1	3
	<u>3</u>	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	<u>6</u>	1		3		3			2			4	

Predizione tramite media pesata :


$$r_{1.5} = (0.41 \cdot 2 + 0.59 \cdot 3) / (0.41 + 0.59) = 2.6$$

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

## CF: Approccio comune

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

- Definiamo la **similitudine**  $s_{ij}$  tra gli oggetti  $i$  e  $j$
- Selezioniamo  $k$  nearest neighbor  $N(i; x)$ 
  - Oggetti più simili a  $i$ , valutati dall'utente  $x$
- Stimiamo il rating  $r_{xi}$  pesato con la media:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$


**Stima baseline per  $r_{xi}$**

$$b_{xi} = \mu + b_x + b_i$$

- $\mu$  = media generale di tutti i film
- $b_x$  = deviazione del rating dalla media dell'utente  $x = (\text{avg. rating utente } x) - \mu$
- $b_i$  = deviazione del rating per l'oggetto  $i$

## Pro e contro

- +: Lavora con tutti i tipi di oggetti**
- +: Non è necessaria feature selection**
- : New user problem**
- : New item problem**
- : Matrice dei rating sparsa**

# Dimensionality Reduction

- Possibile soluzione per il problema dovuto alla **“sparsità” della matrice?**
  - Dimensionality Reduction
- **Latent Semantic Indexing (LSI)**
  - Tecnica algoritmica sviluppata tra la fine degli anni '80 primi anni '90
  - Risolve problemi di sinonima, “sparsità” e scalabilità per grandi dataset
  - Riduce la dimensionalità e cattura le relazioni latenti
- **Si può mappare facilmente nel Collaborative Filtering!**

- Tecnica di indicizzazione che usa la **Singular Value Decomposition** per identificare pattern nella relazione tra termini e concetti contenuti nel testo.
- Si basa sul principio che parole che sono usate nello stesso contesto tendono ad avere significato simile.

- Matrice Termini-Documenti
- Spazio dei concetti
- Mapping tra:

- **Termini  $\leftrightarrow$  Concetti**
- **Documenti  $\leftrightarrow$  Concetti**



- Matrice Utenti-Oggetti
- Spazio delle categorie
- Mapping tra:

- **Oggetti  $\leftrightarrow$  Categorie**
- **Utenti  $\leftrightarrow$  Categorie**

# Part IV

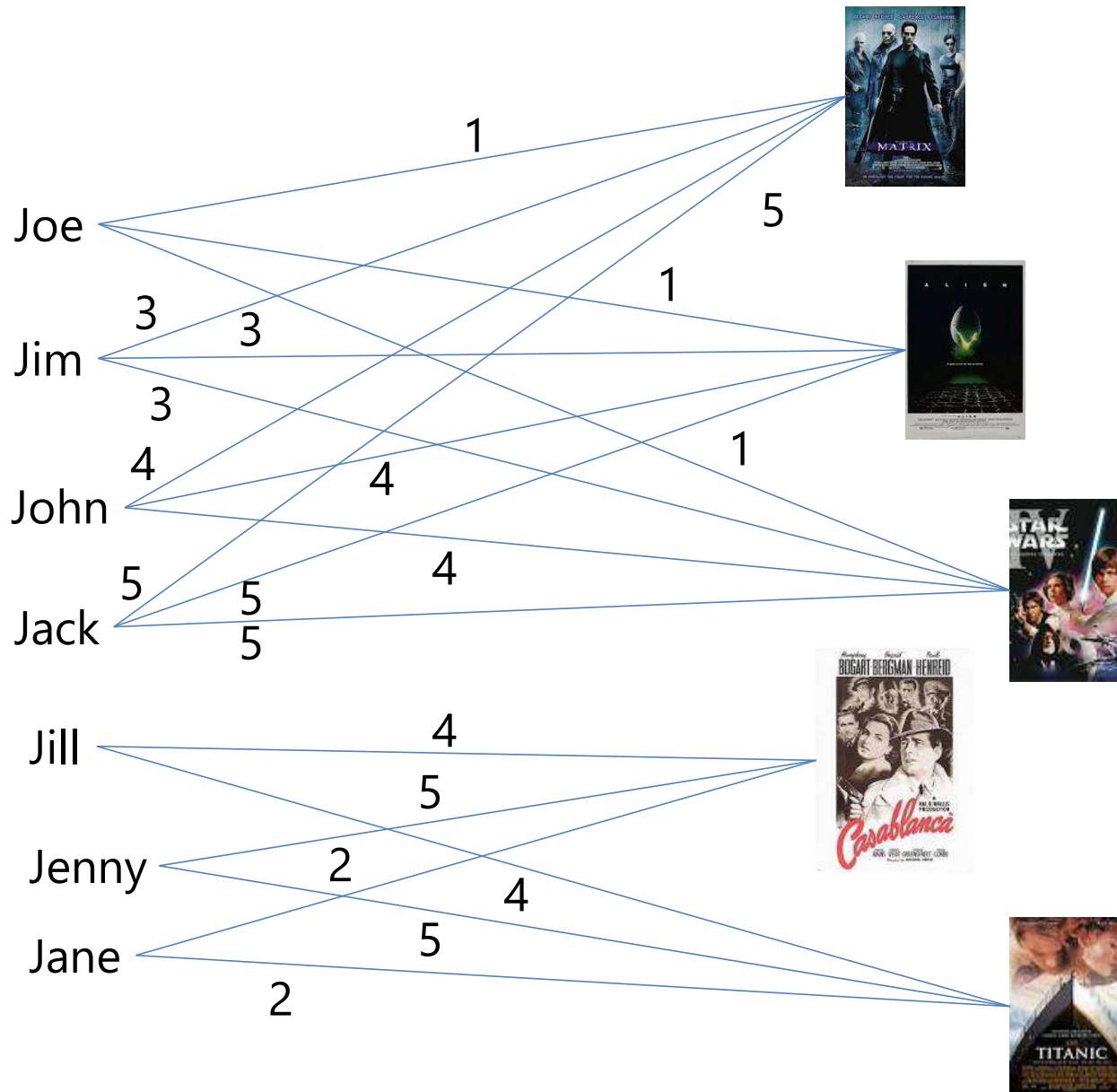
Metodi Graph-based



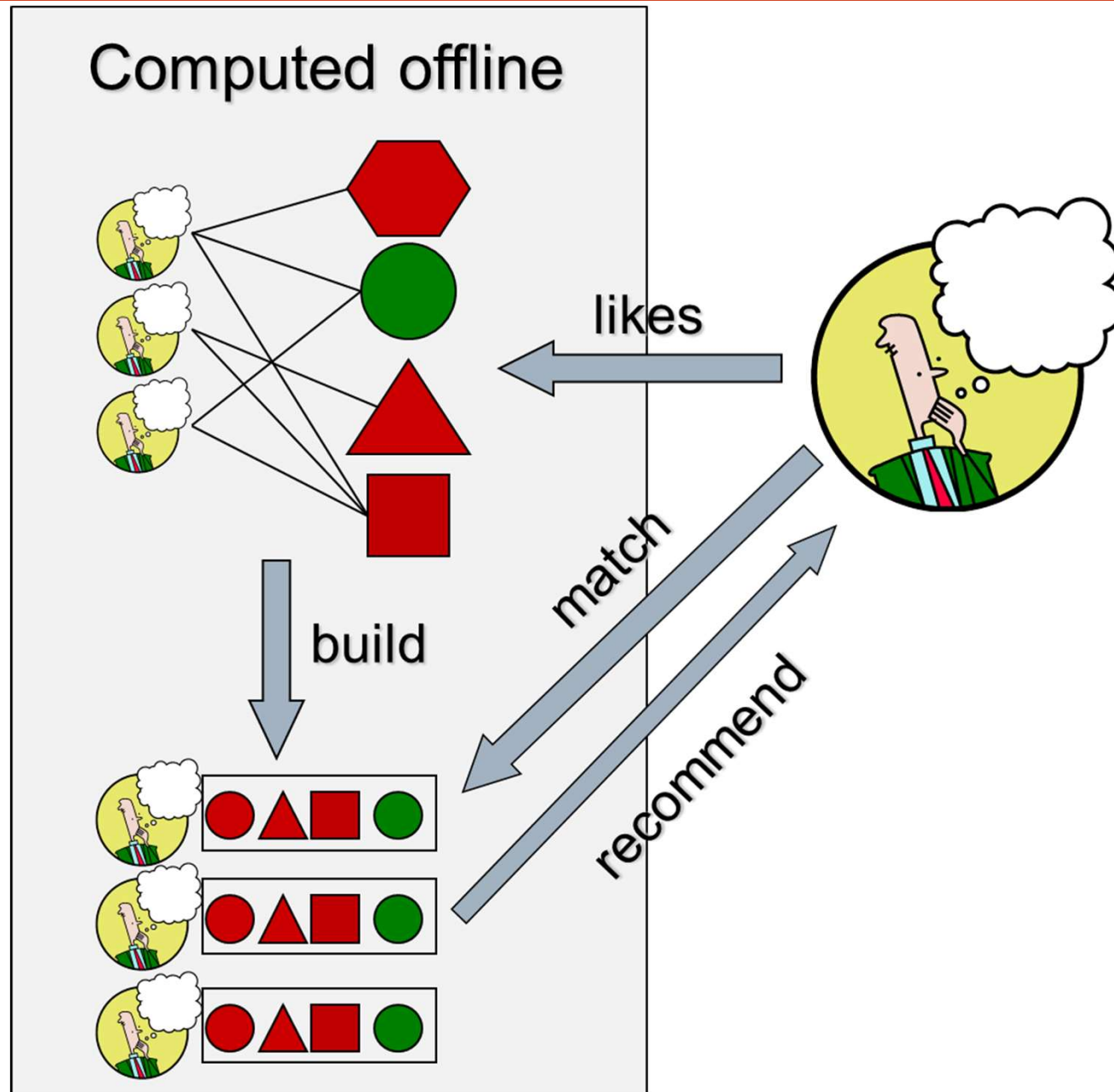
# Metodi Graph-based

- Simile al Collaborative Filtering ma usa un **grafo bipartito** per immagazzinare le informazioni.
- Le raccomandazioni sono ottenute a partire dalla struttura della rete bipartita.
- Due classi di identità: **Utenti (U)** e **Oggetti (O)**.
- Definiamo il grafo bipartito come segue:
  - $G(U, O, E, W)$
  - $E = \{e_{ij} : u_i \text{ likes } o_j\}$
  - $W : E \rightarrow \mathbb{R}$
  - “E” insieme degli archi e “W” è una funzione che rappresenta il degree di preferenza che un utente ha per il particolare oggetto.

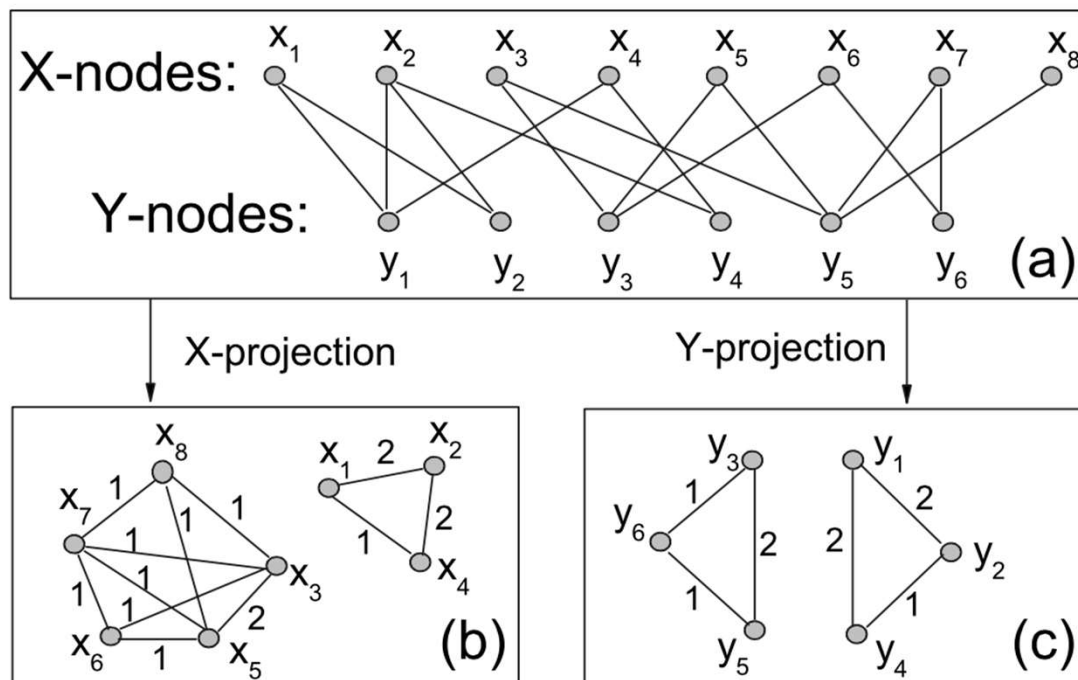
# La utility matrix come grafo bipartito



# Schema del metodo

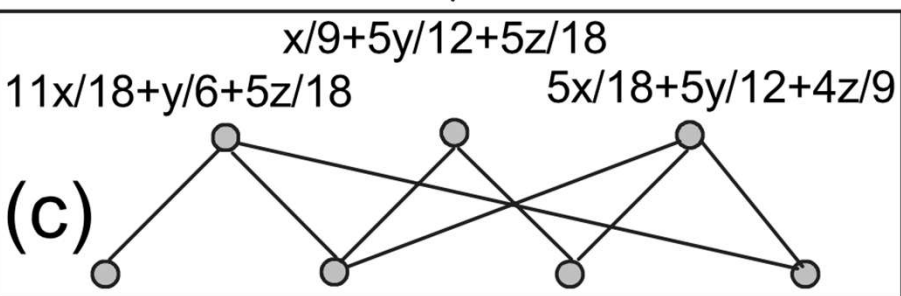
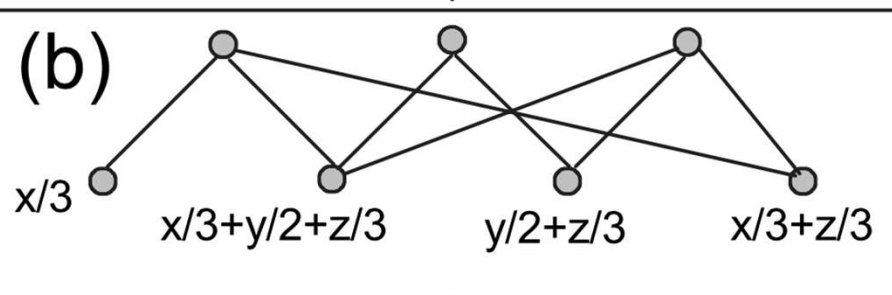
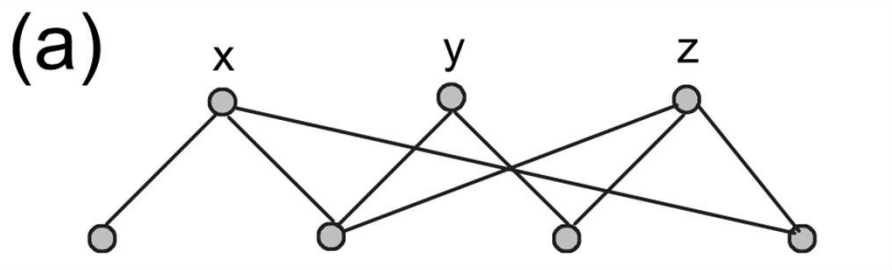


- **NBI (Network-based inference)** sviluppato nel 2007.
- **Bipartite network projection** per ottenere informazioni sulle proiezioni.
- Dopo la proiezione:
  - Reti con nodi dello stesso tipo (utenti o oggetti);
  - Due nodi sono connessi se sono collegati da almeno un nodo di altro tipo;



- Idea base dell'algoritmo: flusso delle risorse sulla rete bipartita:
  - agli oggetti viene assegnata una quantità iniziale di risorse;
  - in un processo a due fasi le risorse sono trasferite dagli oggetti agli utenti e successivamente trasferiti indietro agli oggetti;
  - questo processo, con una fase di normalizzazione consente di ottenere degli score per le coppie utenti - oggetti.

# NBI



- Il calcolo dei pesi avviene attraverso la seguente equazione:

$$w_{ij} = \frac{1}{\Gamma(i, j)} \sum_{l=1} \frac{a_{il} a_{jl}}{D(u_l)}$$

- Dove  $\Gamma(i, j)$  è definita come :

- $\Gamma(i, j) = D(o_j)$  per NBI
- $\Gamma(i, j) = D(o_i)$  per HeatS

- $D(t)$  degree del nodo “t” nella rete bipartita.

## Raccomandazioni con NBI

- Il peso “ $w_{ij}$ ” della proiezione corrisponde a quante risorse vengono trasferite dall’oggetto “ $j$ ” all’oggetto “ $i$ ”, o quanto piacerà l’oggetto “ $j$ ” ad un utente a cui piace l’oggetto “ $i$ ”.
- Data la matrice di adiacenza “ $A$ ” del grafo bipartito e la matrice “ $W$ ”, la matrice di raccomandazione “ $R$ ” per tutti gli utenti può essere calcolata in un unico step:

$$R = A \times W$$

## Pro e contro

- +: Funziona su tutti i tipi di oggetti**
- +: Risolve il problema della sparsità della utility matrix.**
- : New user problem**
- : New item problem**
- : Richiede importanti risorse computazionali**



# Metodi ibridi

- Usare in combinazione diversi metodi di raccomandazione
- Supponiamo di avere i metodi “**X**” e “**Y**” che danno rispettivamente gli score “ **$x_a$** ” e “ **$y_a$** ”. Lo score di un modello ibrido può essere ottenuto come:

$$z_a = (1 - \lambda) \frac{x_a}{\max_{\beta} x_{\beta}} + \lambda \frac{y_a}{\max_{\beta} y_{\beta}}$$

- **$\lambda \in [0;1]$**

## Pro e contro

**+: Efficace nel migliorare la qualità delle raccomandazioni**

**-: Computazionalmente pesante**

# Part VI

## Results Evaluation

# Valutazione delle previsioni

- Confrontare le previsioni con le valutazioni note

- Root-mean-square error (RMSE)

$$RMSE(\hat{Y}, Y) = \sqrt{\frac{\sum_{i=1}^{|\hat{Y}|} (\hat{Y}_i - Y_i)^2}{|\hat{Y}|}}$$

- Un altro approccio: modello 0/1

- Coverage

- Numero di elementi/utenti per i quali il sistema può effettuare previsioni

- Precision/Recall

$$\text{Precision} = \frac{tp}{tp + fp}$$

- Accuracy of predictions

$$\text{Recall} = \frac{tp}{tp + fn}$$

- Receiver operating characteristic (ROC)

- Curva di tradeoff tra falsi positive and falsi negativi

# The Netflix Prize

- **Training data**

- 100 milioni di rating, 480,000 utenti, 17,770 film
- 6 anni di dati: 2000-2005

- **Test data**

- Ultimi rating degli utenti (2.8 million)
- **Criterio di valutazione:** Root Mean Square Error (RMSE)  $= \frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$
- **Netflix's system RMSE: 0.9514**

- **Competizione**

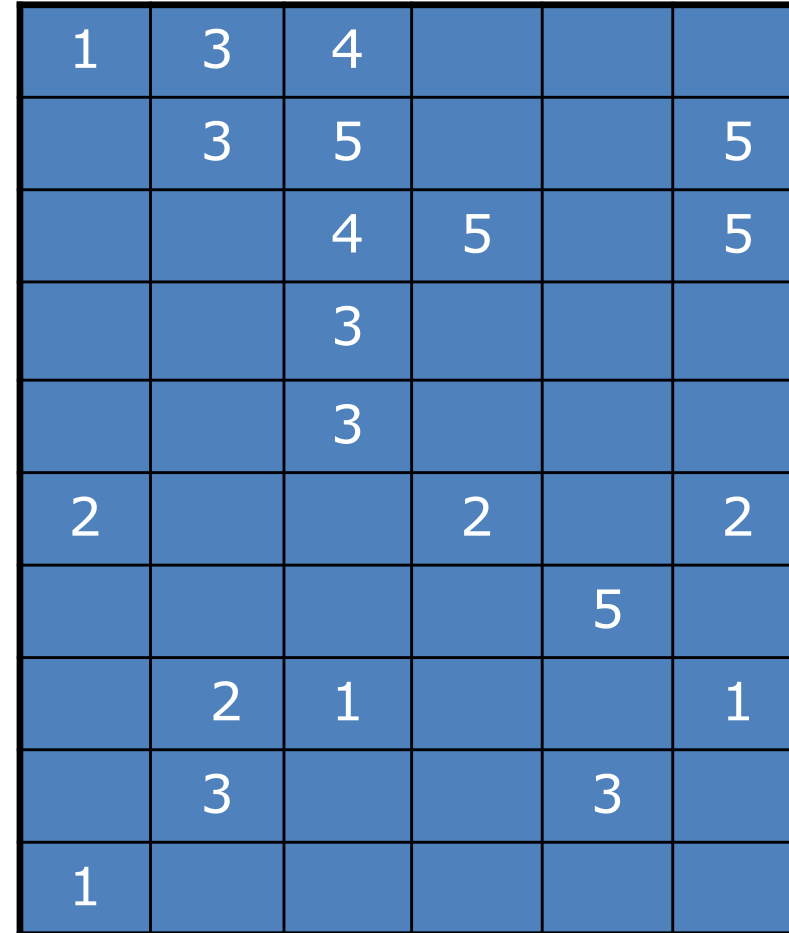
- 2,700+ team
- **\$1 million** prize for 10% improvement on Netflix

# La matrice di utilità Netflix R

***Matrice R***

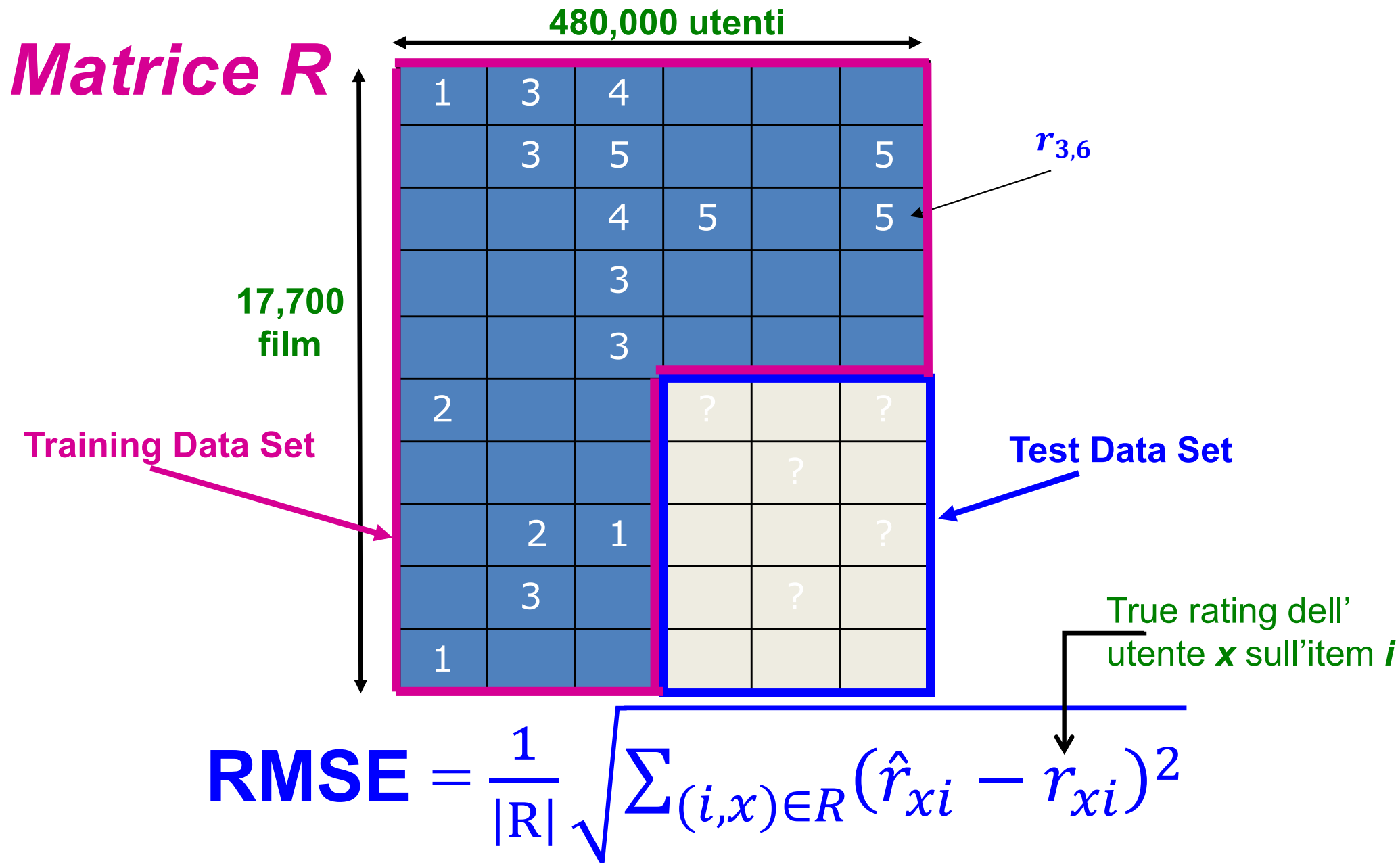
17,700  
film

480,000 utenti



1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

# Matrice di utilità R: Valutazione



# BellKor Recommender System

- **Vincitore della Netflix Challenge!**

- **Multi-scale modeling :**

Combinare top level, “regional”  
modeling dei dati, con una vista locale:

- **Global:**

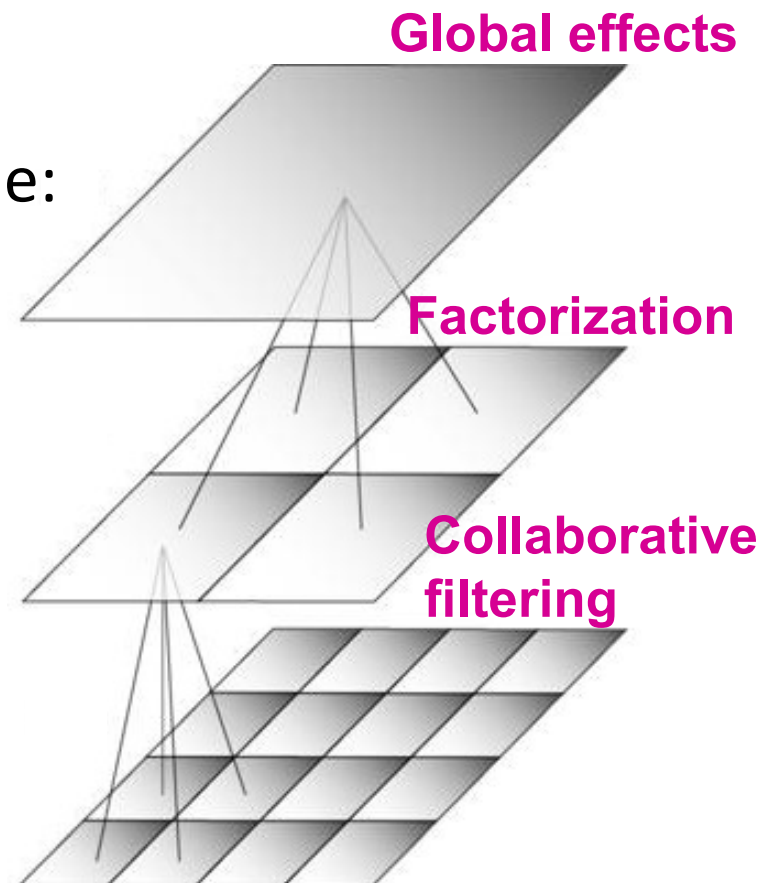
- Overall deviations of users/movies

- **Factorization:**

- Addressing “regional” effects

- **Collaborative filtering:**

- Extract local patterns





# Modellazione di effetti locali e globali

- **Globale:**

- Mean movie rating: **3.7 stelle**
- *Il sesto senso* is **0.5** stelle sopra la media.
- Joe valuta **0.2** stelle sotto la media
- $\Rightarrow$  **Baseline estimation:**

*Joe valuterà **il sesto Senso** 4 stelle*



- **Local neighborhood (CF/NN):**

- A Joe non è piaciuto *Signs*
- $\Rightarrow$  **Final estimate:**

*Joe valuterà **il sesto Senso** 3.8 stelle*



# Riepilogo: Filtraggio collaborativo (CF)

- Il primo e il più popolare metodo **collaborative filtering**
- Ricavare valutazioni sconosciute da quelle di film “**simili**” (variante item-item)
- Definiamo la **misura di similarità**  $s_{ij}$  degli item  $i$  e  $j$
- Seleziona i vicini k-NN, calcola la valutazione
  - $N(i; x)$ : item più simili a  $i$  che sono stati valutati da  $x$

$$\hat{r}_{xi} = \frac{\sum_{j \in N(i; x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i; x)} s_{ij}}$$

$s_{ij}$ ... similarità degli item  $i$  e  $j$   
 $r_{xj}$ ...rating dell'utente  $x$  sull'item  $j$   
 $N(i; x)$ ... set di item simili all'item  $i$  già valutati da  $x$

# Modellazione di effetti locali e globali

- In pratica, otteniamo stime migliori se modelliamo le deviazioni:

$$\hat{r}_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

**Stima baseline per  $r_{xi}$**

$$b_{xi} = \mu + b_x + b_i$$

$\mu$  = Valutazione media complessiva

$b_x$  = Deviazione di valutazione dell'utente  $x$   
= (avg. rating utente  $x$ ) -  $\mu$

$b_i$  = (avg. rating film  $i$ ) -  $\mu$

## Problemi:

- 1) Le misura di simiòarità sono “arbitrarie”
- 2) Le simiarietà pairwise trascurano le interdipendenze tra gli utenti
- 3) Prendere una media ponderata può essere limitante

**Solution:** Invece di  $s_{ij}$  usare  $w_{ij}$  che stimiamo direttamente dai dati

# Idea: Pesi di interpolazione $w_{ij}$

- Usare una **somma pesata** invece di una **media pesata**:

$$\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj})$$

- **Alcune note:**

- $N(i; x)$  ... set di film valutati da x simili al film  $i$
- $w_{ij}$  è il peso dell'interpolazione (un numero reale)
  - Permettiamo:  $\sum_{j \in N(i,x)} w_{ij} \neq 1$
- $w_{ij}$  modella l'interazione tra coppie di filmati  
(non dipende dall'utente x)

# Idea: Pesi di interpolazione $w_{ij}$

- $\hat{r}_{xi} = b_{xi} + \sum_{j \in N(i,x)} w_{ij} (r_{xj} - b_{xj})$

- **Come impostare  $w_{ij}$ ?**

- Ricorda, la metrica di errore è:  $\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2}$  o equivalentemente **SSE**:

$$\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2$$

- Trovare  $w_{ij}$  che minimizzano **SSE sui dati di training!**
  - Models relationships between item  $i$  and its neighbors  $j$
- $w_{ij}$  può essere **appreso/stimato basandosi su**  $x$  and e sugli altri utenti che hanno valutato  $i$

***Perché è una buona idea?***

# Raccomandazioni tramite ottimizzazione

- **Goal: Fare Buone raccomandazioni**

- Quantifica la bontà usando **RMSE**:

**Lower RMSE  $\Rightarrow$  raccomandazioni migliori**

- Desidera fornire buone raccomandazioni su elementi che l'utente non ha ancora visualizzato. **Non posso davvero farlo!**

- **Costruiamo un sistema in modo tale che funzioni bene su valutazioni note (utente, articolo)**

E speriamo che il sistema preveda **bene** anche le valutazioni sconosciute

1	3	4			
	3	5			5
		4	5		5
		3			
		3			
2			2		2
				5	
	2	1			1
	3			3	
1					

# Raccomandazioni tramite ottimizzazione

- Idea: Impostiamo i valori  $w$  in modo che funzionino bene su valutazioni note (utente, articolo)
- Come trovare tali valori  $w$ ?
- Idea: definire una funzione obiettivo e risolvere il problema dell'ottimizzazione
- Trova  $w_{ij}$  che minimizzano l'SSE sui dati di training!

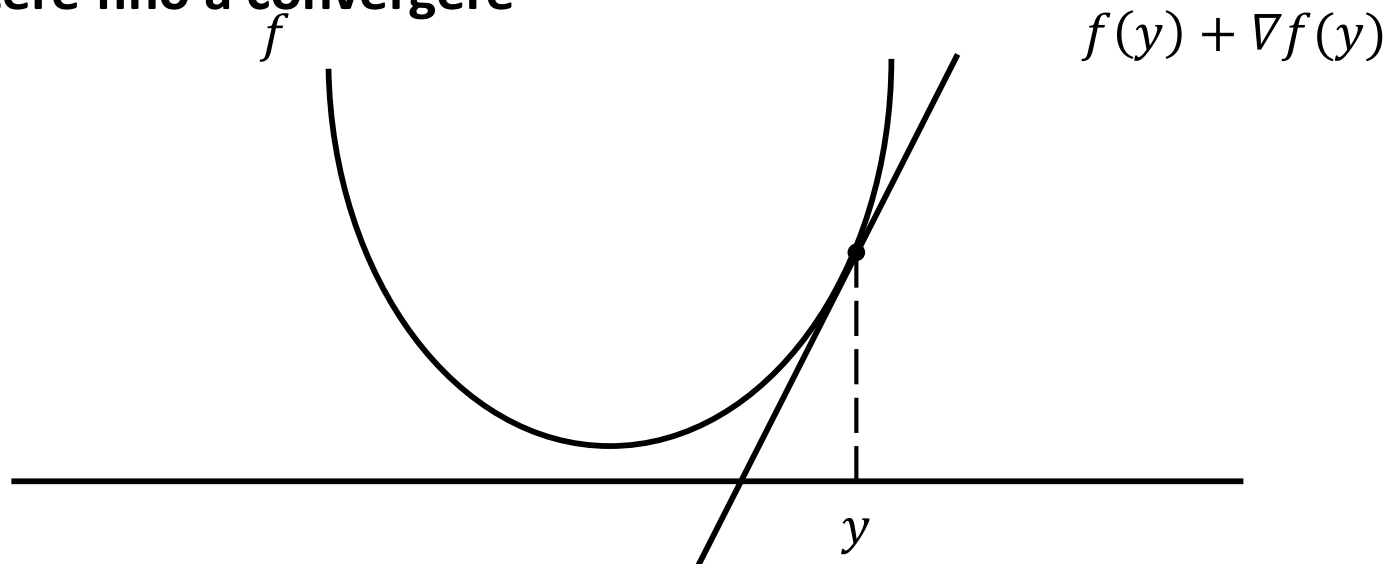
$$J(w) = \sum_{x,i} \left( \underbrace{\left[ b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj}) \right]}_{\text{predizione}} - \underbrace{r_{xi}}_{\text{Rating reale}} \right)^2$$

- Pensa a  $w$  come a un vettore di numeri

# Minimizzare una funzione

- **Un modo semplice per trovare il minimo di una funzione  $f(x)$ :**

- Calcola la derivate  $\nabla f$
- Inizia da un certo punto  $y$  e valuta  $\nabla f(y)$
- Fai un passo nella direzione opposta al gradiente:  $y = y - \nabla f(y)$
- Ripetere fino a convergere





# Pesi di interpolazione

- Abbiamo il problema dell'ottimizzazione, ora cosa facciamo?

$$J(w) = \sum_x \left( \left[ b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj}) \right] - r_{xi} \right)^2$$

- Gradient decent:

- Iterare fino alla convergenza:  $w \leftarrow w - \eta \nabla_w J$
- dove  $\nabla_w J$  è il gradiente (derivate valutata sui data):

$$\nabla_w J = \left[ \frac{\partial J(w)}{\partial w_{ij}} \right] = 2 \sum_{x,i} \left( \left[ b_{xi} + \sum_{k \in N(i;x)} w_{ik} (r_{xk} - b_{xk}) \right] - r_{xi} \right) (r_{xj} - b_{xj})$$

for  $j \in \{N(i; x), \forall i, \forall x\}$

else  $\frac{\partial J(w)}{\partial w_{ij}} = 0$

$\eta$  ... learning rate

while  $|w_{new} - w_{old}| > \epsilon$ :

$w_{old} = w_{new}$

$w_{new} = w_{old} - \eta \cdot \nabla w_{old}$

- **Nota:** Fissiamo il film  $i$ , Andiamo su tutti i  $r_{xi}$ , per ogni film  $j \in N(i; x)$ , calcoliamo  $\frac{\partial J(w)}{\partial w_{ij}}$

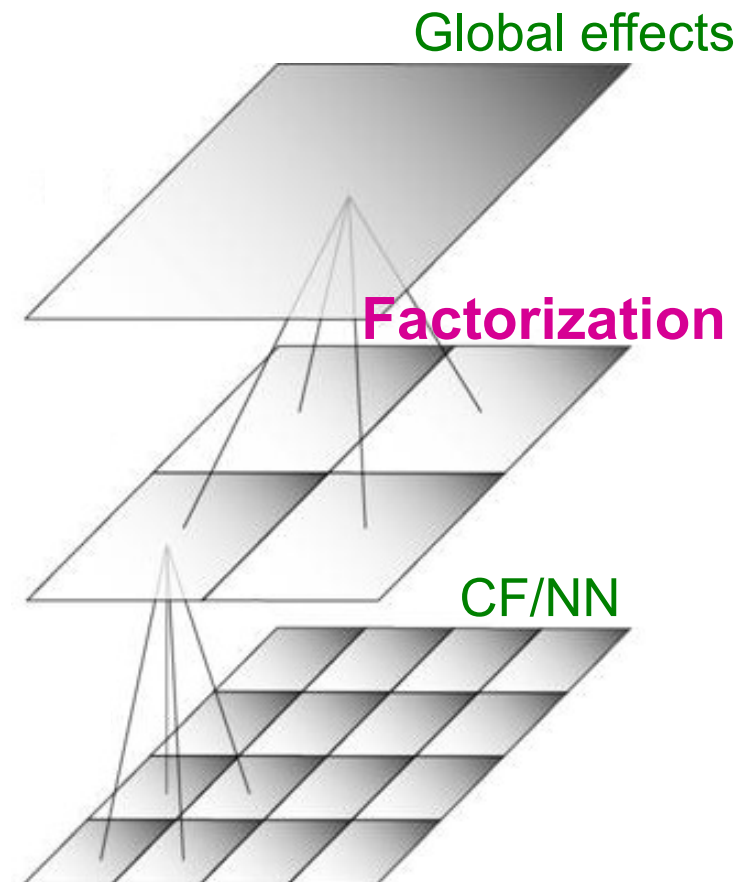
# Pesi di interpolazione

- **Finora:**  $\widehat{r_{xi}} = b_{xi} + \sum_{j \in N(i;x)} w_{ij} (r_{xj} - b_{xj})$

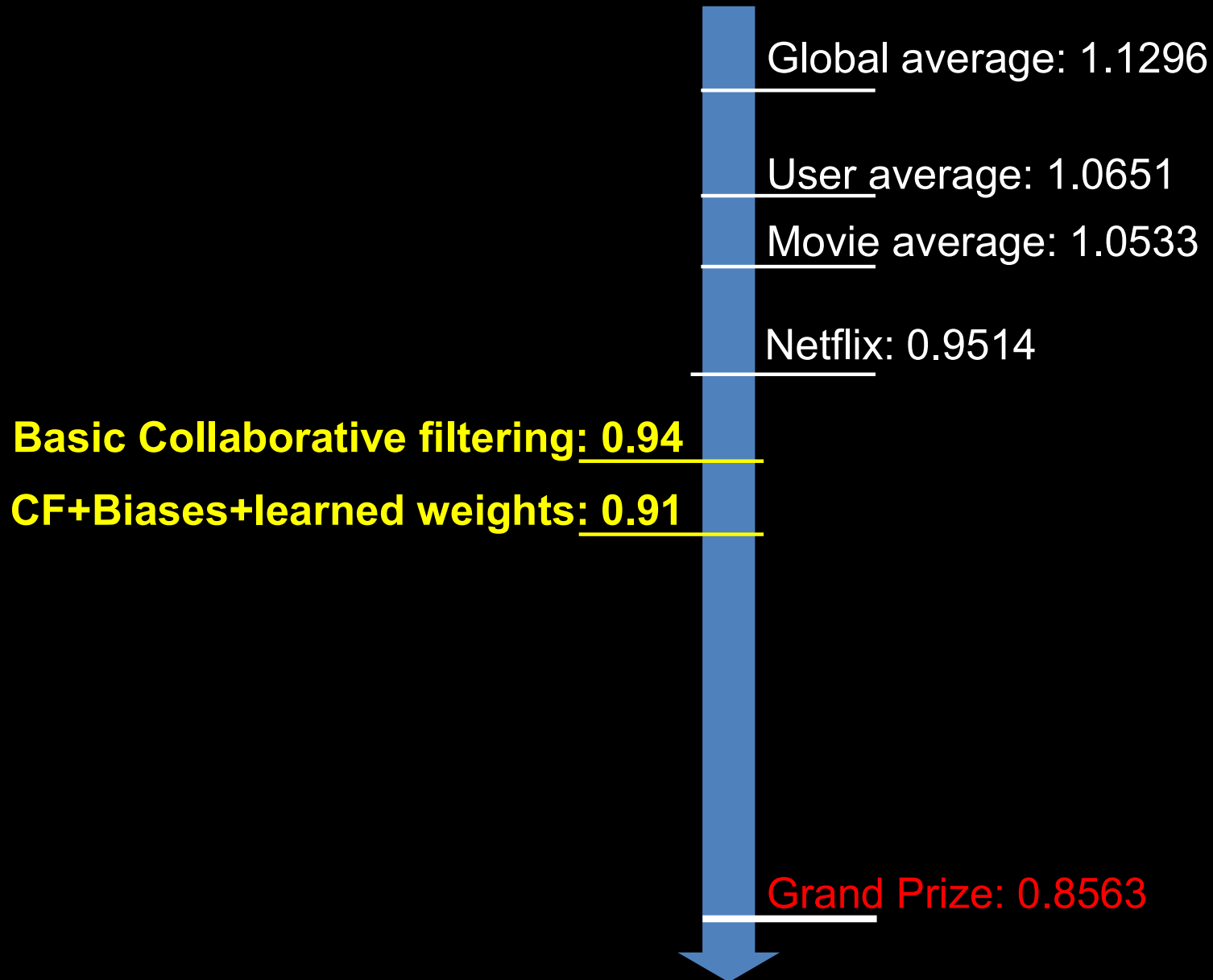
- Pesi  $w_{ij}$  derivato basato sul loro ruolo; **l'assenza di ricorso a una misura arbitraria di similarità** ( $w_{ij} \neq s_{ij}$ )
- Tenere conto in modo esplicito delle interrelazioni tra i film vicini

- **Prossimo: Latent factor model**

- Estrarre “regional” Correlazioni



# Performance dei vari metodi



# Esercizio

- Considerare il modello NBI:

$$w_{ij} = \frac{1}{\Gamma(i,j)} \sum_{l=1} \frac{a_{il}a_{jl}}{D(u_l)}$$

Generalizzarlo definendo  $\frac{1}{\Gamma(i,j)}$  come un peso  $\gamma_{ij}$

$$\gamma = \begin{bmatrix} \gamma_{1,1} & \cdots & \gamma_{1,n} \\ \vdots & \ddots & \vdots \\ \gamma_{n,1} & \cdots & \gamma_{n,n} \end{bmatrix}$$

$$w_{ij} = \gamma_{ij} \sum_{l=1} \frac{a_{il}a_{jl}}{D(u_l)} \quad r_{xi} = \sum_j a_{xj} \times w_{ji} \rightarrow r_{xi} = \sum_j a_{xj} \times \left[ \gamma_{ij} \sum_{l=1} \frac{a_{il}a_{jl}}{D(u_l)} \right]$$

$$\frac{1}{|R|} \sqrt{\sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2} \text{ consideriamo l'SSE } \sum_{(i,x) \in R} (\hat{r}_{xi} - r_{xi})^2$$

$$\min_{\gamma} \sum_{(i,x) \in R} \left( \left( \sum_j a_{xj} \times \left[ \gamma_{ij} \sum_{l=1} \frac{a_{il}a_{jl}}{D(u_l)} \right] \right) - r_{xi} \right)^2$$

Definire  $J(\gamma)$ . Ottimizzare i pesi  $\gamma_{ij}$  utilizzando il Gradiente Discendente:

# Notebook

<https://colab.research.google.com/drive/1taAEyeHeNrddjYO3niJSrRR9sduYnt1?usp=sharing>