

# Principal-Component Analysis

- **Principal-Component Analysis**, o **PCA**, è una tecnica che prende un dataset relativo ad un insieme di tuple in uno spazio ad alta dimensione e trova le direzioni lungo il quale le tuple si allineano meglio.
- Trattiamo l'insieme di tuple come una matrice  $\mathbf{M}$  e troviamo gli autovettori di  $\mathbf{MM}^T$  o  $\mathbf{M}^T\mathbf{M}$ .
- La matrice di questi autovettori possono essere pensati come una rotazione rigida dello spazio ad alta dimensione.

# PCA

- Algoritmo PCA:

1.  $M \leftarrow$  Crea una matrice di dati  $N \times d$ , con ogni riga un vettore riga  $m_n$  dei dati
2.  $M \leftarrow$  sottraiamo la media  $m$  da ogni vettore riga  $m_n$  in  $M$
3.  $\Sigma \leftarrow$  matrice di covarianza di  $M$
4. Trova gli autovalori e autovettori di  $\Sigma$

- PC's  $\leftarrow$  gli X autovettori con I piu' grandi autovalori

# PCA

Trasforma insieme di dati in uno spazio a dimensione ridotta

↳ Cerca di preservare la migliore parte delle variazioni presenti nei dati originali

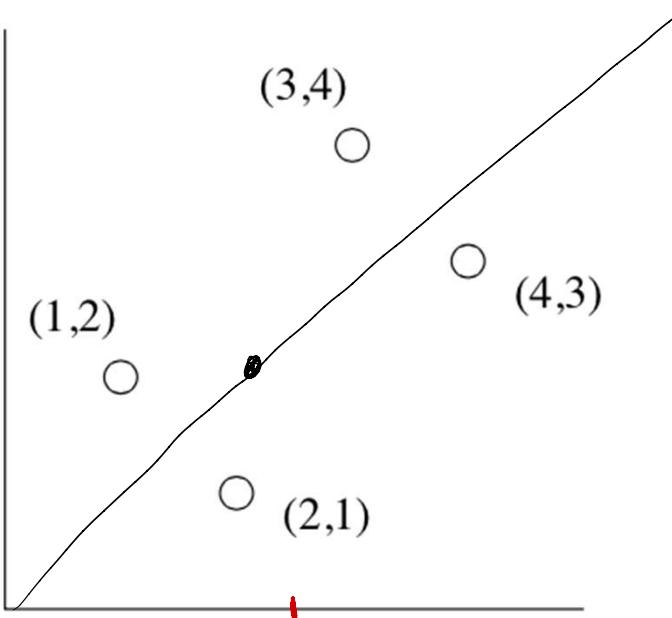
Troviamo insieme di t-uple come la matrice  $M$   
Troviamo autovettori di  $MM^T$  o  $M^T M$

Matrici autovettori possono essere pensati come coordinate rigide dello spazio ad alte dimensioni.

## Algoritmo PCA

1)

1.  $M \leftarrow$  Crea una matrice di dati  $N \times d$ , con ogni riga un vettore riga  $m_n$  dei dati
  2.  $M \leftarrow$  sottraiamo la media  $m$  da ogni vettore riga  $m_n$  in  $M$
  3.  $\Sigma \leftarrow$  matrice di covarianza di  $M$
  4. Trova gli autovalori e autovettori di  $\Sigma$
- PC's  $\leftarrow$  gli  $X$  autovettori con i più grandi autovalori



$$M = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix}$$

$$M^T M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} = \begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix}$$

Trova gli autovalori

$$(30 - \lambda)(30 - \lambda) - 28 \times 28 = 0$$

$$\lambda = 58 = \lambda_1 \rightarrow \text{primo autovalore}$$

$$\lambda = 2 = \lambda_2$$

I corrispondenti autovettori

$$\begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 58 \begin{bmatrix} x \\ y \end{bmatrix} \quad \begin{bmatrix} 30 & 28 \\ 28 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 2 \begin{bmatrix} x \\ y \end{bmatrix}$$

$$30x + 28y = 58x$$

$$28x + 30y = 58y$$

$$\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} \text{ Autovettore}$$

$$\begin{vmatrix} 30-\lambda & 28 \\ 28 & 30-\lambda \end{vmatrix}$$

Costruisci **E**, matrice degli autovettori di  $M^T M$ . Mettere per primo il primo autovettore

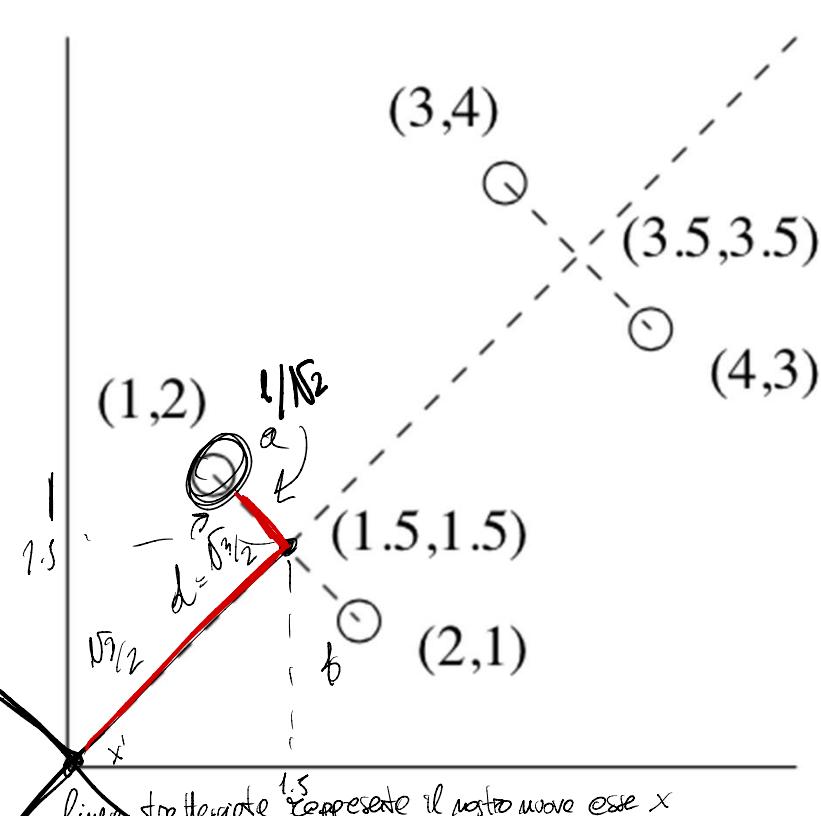
Proiezione dati

$$E = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$ME = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3/\sqrt{2} & 1/\sqrt{2} \\ 3/\sqrt{2} & -1/\sqrt{2} \\ 7/\sqrt{2} & 1/\sqrt{2} \\ 7/\sqrt{2} & -1/\sqrt{2} \end{bmatrix}$$

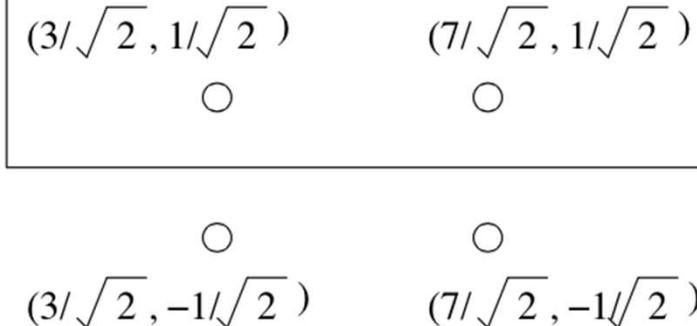
2.12 0.70

1.3



linea che rappresenta il nostro nuovo espace  
Punto più vicino all'asse lo colloca a distanza  $\sqrt{2}/2$  dall'asse  
Proiezione punto sull'asse è quella di  $a$  e  $b$  è  $1.5, 1.5$   
LA DISTANZA TRA ORIGINI E OGNI PUNTO È  

$$\sqrt{(1.5)^2 + (1.5)^2} = \sqrt{4.5} = 3/\sqrt{2}$$



- **ME** è il punto di **M** trasformato in uno spazio di nuove coordinate. In questo spazio, **il primo asse (quello che corrisponde al piu' grande autovalore)** è il più significativo; formalmente, la **varianza** di un punto lungo quest'asse **è la più grande**.
- Il secondo asse, che corrisponde al secondo autovalore, è il successivo secondo autovalore più significativo nello stesso senso. Questo pattern si presenta per ogni autocoppia.

- Per **trasformare M** in uno spazio con meno dimensioni: **Preserviamo le dimensioni che usano gli autovettori associati ai piu' alti autovalori e cancelliamo gli altri**

*Aut. Val. → Trade*

$$E = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 3 & 4 \\ 4 & 3 \end{bmatrix} \times \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3/\sqrt{2} & 3/\sqrt{2} \\ 3/\sqrt{2} & 7/\sqrt{2} \\ 7/\sqrt{2} & 7/\sqrt{2} \end{bmatrix}$$

# Notebook

<https://colab.research.google.com/drive/1lykoVbdYyHVvdJ7dUme5yN2wNGbQOrQV?usp=sharing>

# Singular Value Decomposition (SVD)

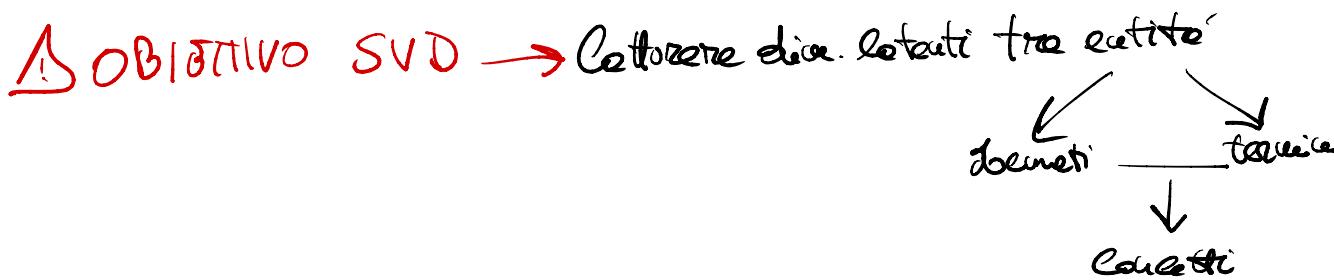
# SVD - Definizione

$$\mathbf{A}_{[m \times n]} = \mathbf{U}_{[m \times r]} \Sigma_{[r \times r]} (\mathbf{V}_{[n \times r]})^T$$

- **A: Matrice dei dati di input** → può essere sempre decomposta in 3 matrici
  - Matrice  $m \times n$  (e.g.,  $m$  documenti,  $n$  termini)
- **U: Vettori singolari di sinistra**
  - Matrice  $m \times r$  matrix ( $m$  documenti,  $r$  concetti)  
oppure concetti in termini di documenti
- **$\Sigma$ : Valori singolari** → indica quanto ciascun concetto è importante o rilevante per spiegare la varianza nei dati in input
  - Matrice  $r \times r$  diagonale (forza di ogni ‘concetto’)  
( $r$  : rango di  $\mathbf{A}$ )
- **V: Vettori singolari di destra**
  - Matrice  $n \times r$  ( $n$  termini,  $r$  concetti)  
oppure concetti in termini di termini

Rolle simile per SVD e PCA

Concetti → dimensioni nello spazio latente



Nel contesto della **Singular Value Decomposition (SVD)** trattata nelle fonti, i **concetti** rappresentano delle **dimensioni latenti o fattori latenti** sottostanti ai dati originali. Quando una matrice viene decomposta tramite SVD in tre matrici ( $U$ ,  $\Sigma$ ,  $V^T$ ), le colonne della matrice  $U$  possono essere interpretate come una base vettoriale che definisce uno spazio di **concetti**. In questo spazio, le righe della matrice originale (che potrebbero rappresentare documenti, utenti, ecc.) vengono proiettate.

Più in dettaglio:

- La matrice originale può essere pensata come una relazione tra due insiemi di entità (ad esempio, documenti e termini, o utenti e film).

- L'SVD cerca di identificare dei **temi o categorie nascoste** che spiegano le relazioni osservate. Questi temi o categorie sono i **concetti**.

- Nella matrice  $U$  (*matrice dei vettori singolari sinistri*), ogni colonna rappresenta un **concetto**. Le righe di  $U$  indicano il grado in cui ciascuna delle entità del primo insieme (ad esempio, un utente) è associata a ciascun **concetto**.

- Nella matrice  $V^T$  (*trasposta della matrice dei vettori singolari destri*), ogni riga rappresenta un **concetto**, e le colonne indicano il grado in cui ciascuna delle entità del secondo insieme (ad esempio, un film) è associata a ciascun **concetto**.

- La matrice diagonale  $\Sigma$  (*matrice dei valori singolari*) contiene valori che indicano la **forza o l'importanza di ciascun concetto**. Valori singolari più grandi corrispondono a concetti che catturano una maggiore variabilità nei dati.

Nell'esempio pratico fornito, dove la matrice rappresenta i rating dati da utenti a dei film, i **concetti** identificati tramite SVD potrebbero essere interpretati come i **generi dei film**, come la "science fiction" e il "romance". La matrice  $U$  indicherebbe quanto ogni utente è interessato a ciascun genere, la matrice  $V^T$  indicherebbe quanto ogni film è rappresentativo di ciascun genere, e la matrice  $\Sigma$  indicherebbe quanto sono "forti" o prevalenti questi generi all'interno dei dati dei rating.

In sintesi, i **concetti** nella SVD rappresentano delle **astrazioni o temi latenti** che emergono dalla struttura dei dati e che permettono di comprendere le relazioni tra le entità in uno spazio di dimensionalità potenzialmente ridotta. Questi concetti non sono esplicitamente presenti nei dati originali, ma vengono inferiti attraverso la decomposizione matriciale.

SVD  $\rightarrow$  SINGOLARE UTENTI / FILM

documenti

utenti

relazione tra utenti e film

RATING INPUT  $\rightarrow$  INTEGRAZIONI OBSERVATE TRA UTENTI E FILM  $\rightarrow$  "rating"

SVD  $\rightarrow$  Decomporre matrice

Comb. lineari di utenti e film  
Cellule non poterne nei dati

Due lettori  
Eseguo la decomposizione

Corretto

Scoprire corretti

metametico

quanto ogni utente  
è associato a ciascuna  
corretto?

Rilevare struttura sottostante

Indice importanza  
Ciascuna corretto  
Cellula variabilità  
dei dati originali

quanto ogni  
film è associato  
e ciascun corretto

interpretativo

## Come scopriamo quali sono i concetti?

Scoprire i concetti significa interpretare le dimensioni latenti che emergono dalla SVD. Questo processo è in parte matematico e in parte interpretativo. Ecco i passaggi pratici:

### a) Calcola la SVD e scegli il numero di concetti ( $k$ )

Dopo aver calcolato la SVD, la matrice  $\Sigma$  contiene i valori singolari in ordine decrescente. Questi valori indicano l'importanza di ciascun concetto.

Decidi quante dimensioni latenti ( $k$ ) vuoi mantenere. Questo si basa su un compromesso:

- Se prendi troppe dimensioni, includi rumore.
- Se ne prendi troppo poche, perdi informazioni utili.

Un metodo comune è guardare la varianza spiegata: calcoli la somma dei quadrati dei valori singolari e scegli  $k$  in modo da spiegare, ad esempio, il 90% della varianza totale.

### b) Analizza le matrici $U$ e $V$ per ogni concetto

Ogni concetto è rappresentato da:

- Una colonna di  $U$ : mostra quanto ogni utente è associato a quel concetto.
- La corrispondente colonna di  $V$ : mostra quanto ogni film è associato a quel concetto.

Per scoprire cosa rappresenta un concetto:

- Guarda i valori alti in  $U$ : Identifica gli utenti con i valori più alti nella colonna di  $U$  corrispondente a quel concetto. Questi utenti sono i più “rappresentativi” di quel concetto.
- Guarda i valori alti in  $V$ : Identifica i film con i valori più alti nella colonna corrispondente di  $V$ . Questi film sono i più “rappresentativi” di quel concetto.

Metti insieme le informazioni: Cerca di capire cosa hanno in comune gli utenti e i film associati a quel concetto.

### c) Interpreta semanticamente i concetti

Questa è la parte più “umana” del processo. La SVD ti dà i numeri (i valori in  $U$  e  $V$ ), ma sta a te dare un significato ai concetti. Per farlo:

- Analizza i film associati: Se i film con valori alti in  $V$  per un concetto sono “Die Hard”, “Rambo” e “Terminator”, potresti interpretare il concetto come “film d’azione”.
- Analizza gli utenti associati: Se gli utenti con valori alti in  $U$  per lo stesso concetto tendono a dare voti alti a quei film d’azione, questo rafforza l’interpretazione.
- Considera il contesto: Nel tuo caso, i concetti potrebbero essere generi cinematografici, ma in un caso reale potrebbero essere più astratti (es. “film che piacciono agli adolescenti” o “film con trame complesse”).

#### **d) Valida l'interpretazione (opzionale)**

Puoi verificare se la tua interpretazione ha senso confrontando i risultati con dati esterni. Ad esempio:

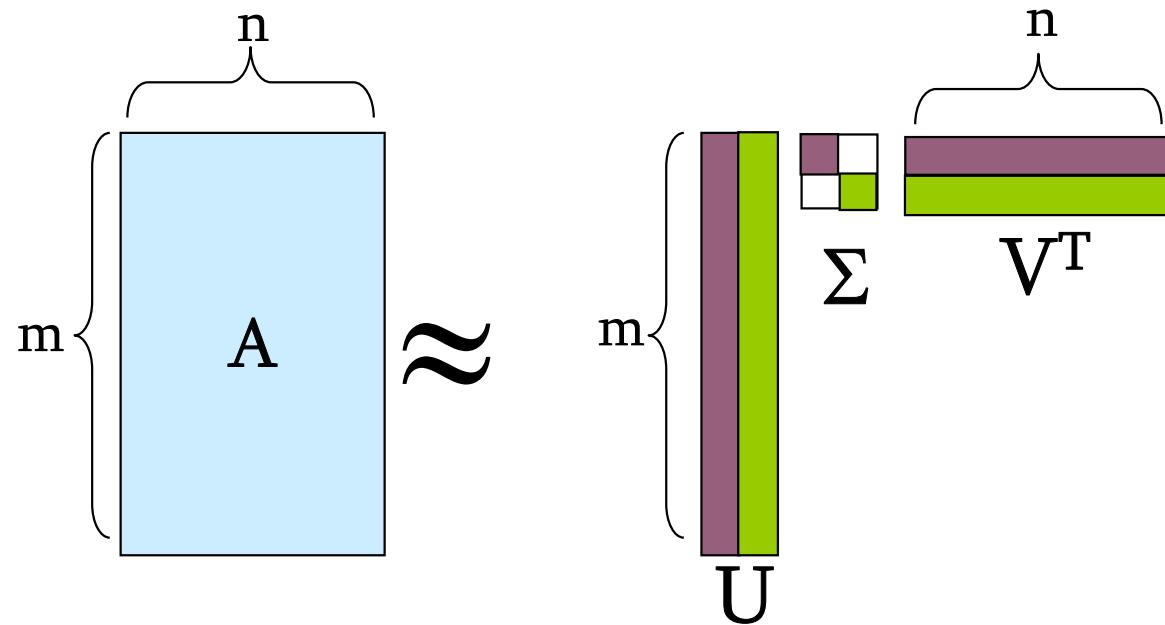
- Se hai metadati sui film (es. il genere ufficiale), puoi controllare se i film associati a un concetto appartengono effettivamente a un genere comune.
- Puoi anche fare un test pratico: usa i concetti per fare raccomandazioni e vedi se gli utenti trovano le raccomandazioni pertinenti.

# SVD

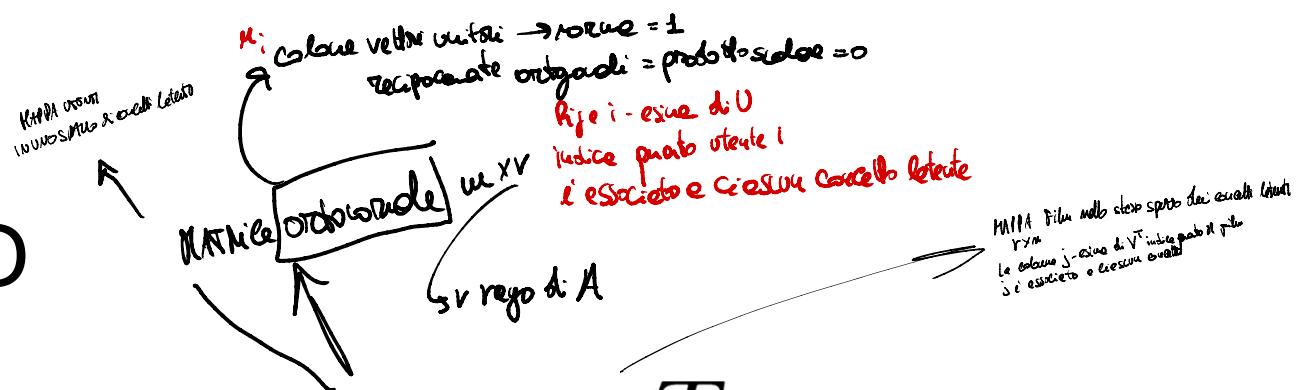
$$A \approx U\Sigma V^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^T$$

Annotations for the SVD components:

- $\mathbf{U}$ : i-th column  $\mathbf{u}_i$
- $\Sigma$ : i-th diagonal value  $\sigma_i$
- $\mathbf{V}^T$ : i-th row of  $\mathbf{V}^T$



SVD



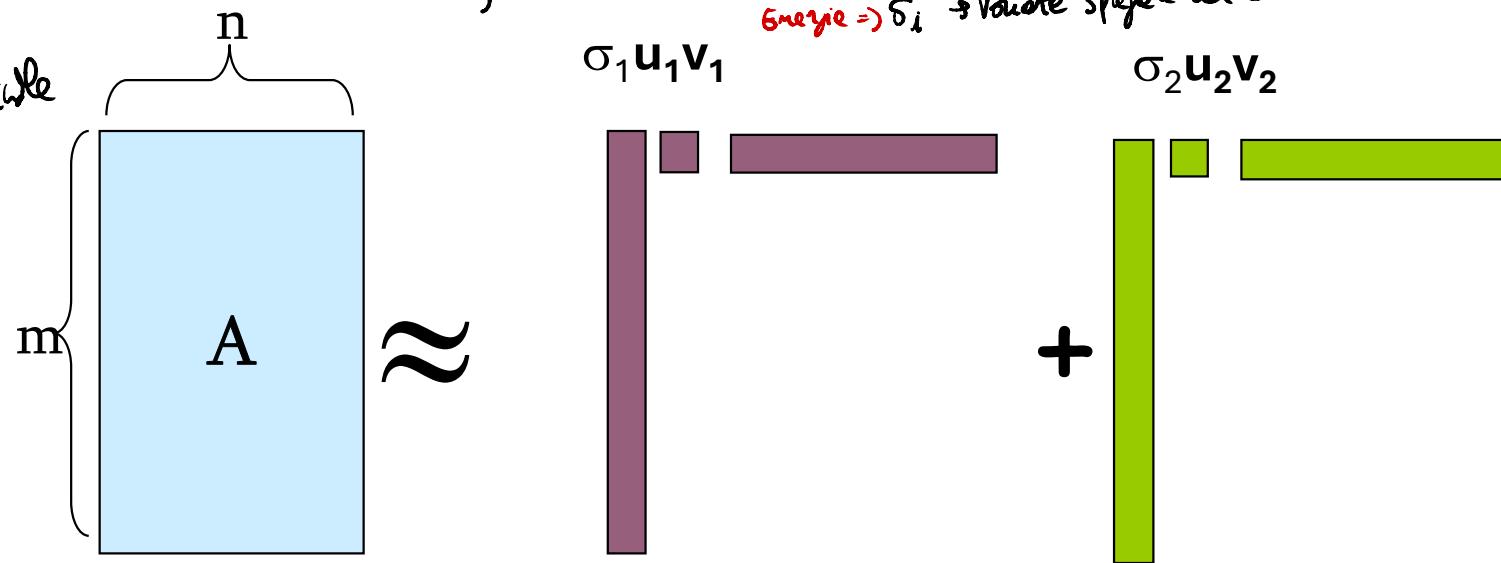
$$A \approx U\Sigma V^T = \sum_i \sigma_i u_i \circ v_i^T$$

Misura importanza di ciascun contributo alla tante  
 $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$   
 Energia  $\Rightarrow \sigma_i^2 \rightarrow$  Variante spiegata dal contributo  $i$ -esimo

$$\sigma_1 u_1 v_1$$

$$\sigma_2 u_2 v_2$$

Matrice input originale



$\sigma_i$  ... scalar  
 $u_i$  ... vector  
 $v_i$  ... vector

Per una data matrice  $A$  esiste sempre  $U \Sigma V^T$  che soddisfa SVD

## SVD - Proprietà

E' **sempre** possibile decomporre una matrice reale  $A$  in  $A = U \Sigma V^T$ , dove

- $U, \Sigma, V$ : unici
- $U, V$ : Ortonormali
  - $U^T U = I; V^T V = I$  ( $I$ : matrice identità)
  - (Le colonne sono vettori unitari ortogonali)
- $\Sigma$ : diagonale
  - Entry (valori singolari) sono positivi, e ordinati in modo decrescente ( $\sigma_1 \geq \sigma_2 \geq \dots \geq 0$ )

Colonne di vettori unitari  
ortogonali fra loro

$$\begin{matrix} \star & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 2 \end{matrix}$$

valori singolari di  $A$

matrice  $A$  è  
i valori singolari positivi

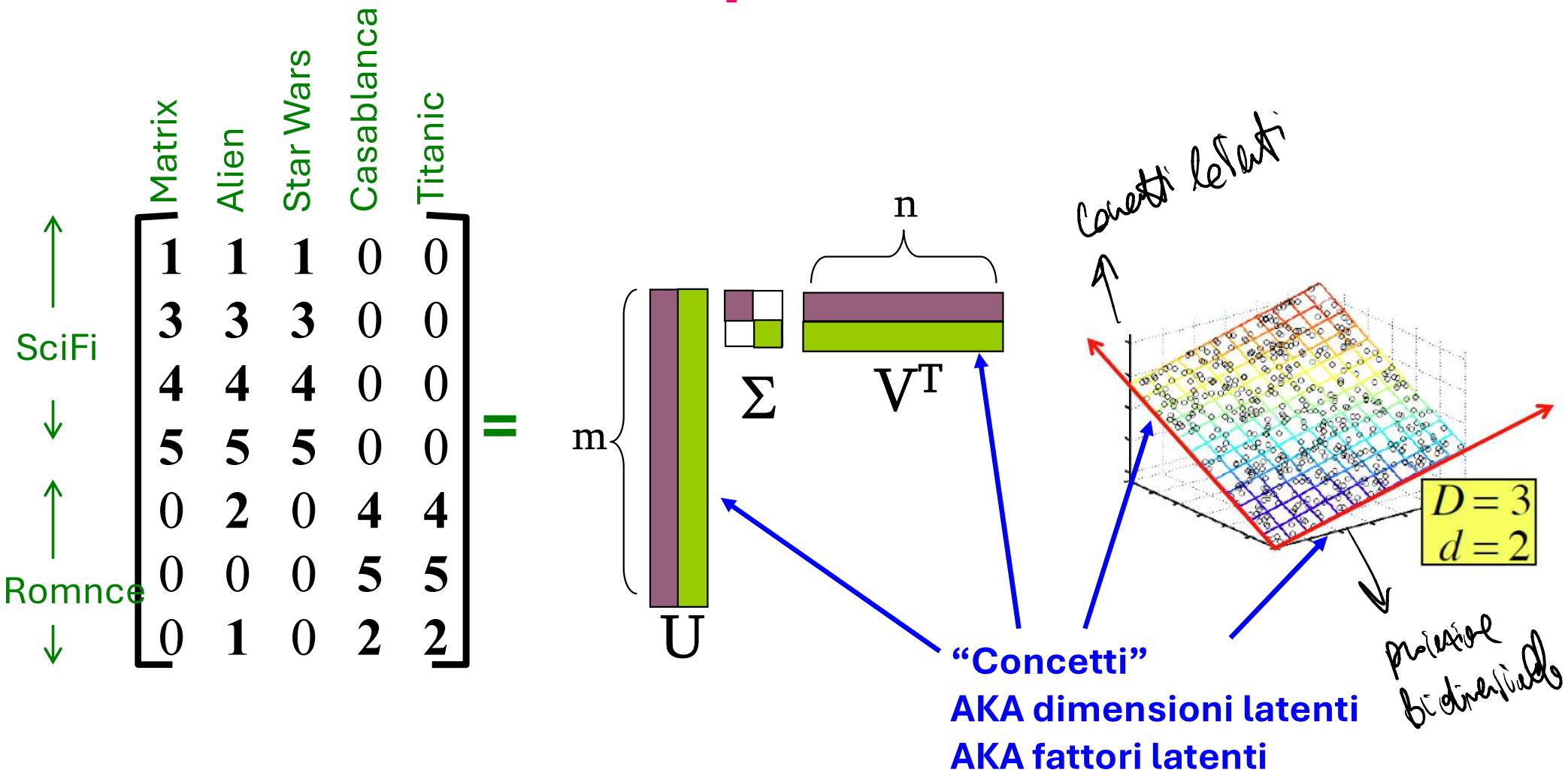
Valori singolari rappresentano posse o importanza di ciascun buotto latente elettro delle componenti

Valore singolare grande  $\rightarrow$  dimensione di buotto che spiega la maggior varianza nei dati originali

↳ riduzione delle dimensioni

# SVD – Esempio: utenti-film

•  $A = U \Sigma V^T$  - Esempio: utenti-film



# SVD – Esempio: utenti-film

•  $A = U \Sigma V^T$  - Esempio: utenti-film

$$\begin{matrix}
 & \begin{matrix} \text{Matrix} & \text{Alien} & \text{Star Wars} & \text{Casablanca} & \text{Titanic} \end{matrix} \\
 \begin{matrix} \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romance} \\ \downarrow \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} & = & \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} & \times & \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} & \times \\
 & & & & & & & \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}
 \end{matrix}$$

# SVD – Esempio: utenti-film

•  $A = U \Sigma V^T$  - esempio: utenti-film

$$\begin{array}{c}
 \begin{array}{c} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \end{array} \\
 \begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{array}
 \end{array}
 \xrightarrow{\quad \quad \quad = \quad \quad \quad}
 \begin{array}{c}
 \begin{array}{c} \text{Concetto-SciFi} \\ \text{Concetto-Romance} \end{array} \\
 \begin{array}{ccc} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{array}
 \end{array}
 \times
 \begin{array}{c}
 \begin{array}{c} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{array} \\
 \times
 \end{array}$$

# SVD – Esempio: utenti-film

•  $A = U \Sigma V^T$  - esempio:

**U Matrice di similarità  
“utenti-concetti”**

	Matrix	Alien	Star Wars	Casablanca	Titanic
SciFi	1	1	1	0	0
↓	3	3	3	0	0
Romance	4	4	4	0	0
↓	5	5	5	0	0
0	2	0	4	4	
0	0	0	5	5	
0	1	0	2	2	

$$= \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix}$$

Concetto-SciFi   Concetto-Romance

$$\times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times$$

$$\begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

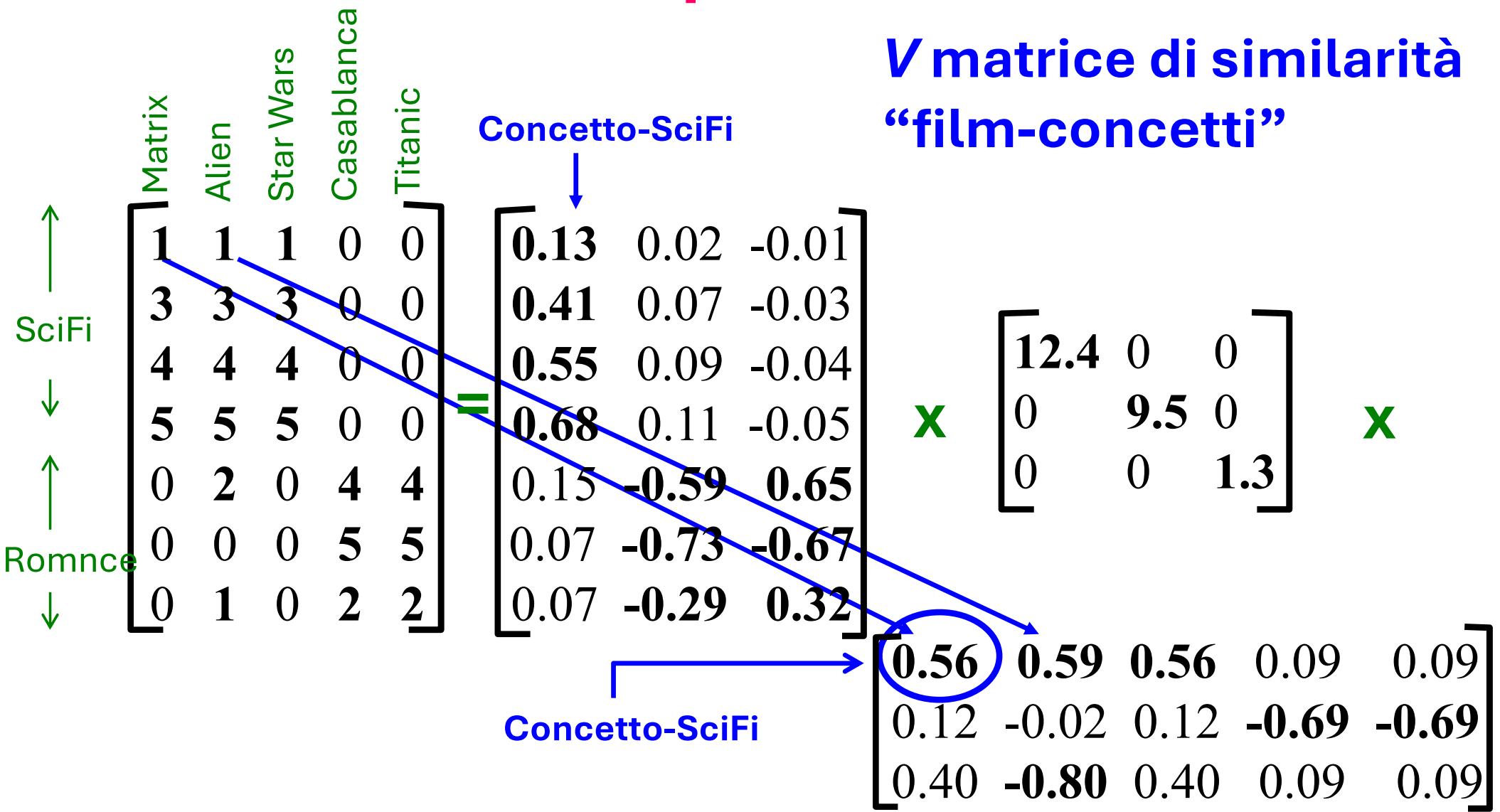
# SVD – Esempio: utenti-film

•  $A = U \Sigma V^T$  - esempio:

$$\begin{array}{c}
 \begin{array}{c}
 \begin{array}{c} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \end{array} \\
 \begin{array}{c} \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romance} \\ \downarrow \end{array}
 \end{array} \\
 \begin{array}{c}
 \begin{array}{c} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \end{array} \\
 \begin{array}{c} \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romance} \\ \downarrow \end{array}
 \end{array}
 \end{array}
 = \begin{array}{c}
 \begin{array}{c} \text{Concetto-SciFi} \\
 \downarrow
 \end{array} \\
 \begin{array}{c} \begin{array}{ccc} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{array} \end{array}
 \end{array}
 \begin{array}{c}
 \begin{array}{c} \text{"forza" del Concetto-SciFi} \\
 \downarrow
 \end{array} \\
 \begin{array}{c} \begin{array}{ccc} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{array} \end{array}
 \end{array}
 \times \begin{array}{c}
 \begin{array}{c} \text{X} \end{array} \\
 \begin{array}{c} \begin{array}{ccccc} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{array} \end{array}
 \end{array} \times$$

# SVD – Esempio: utenti-film

•  $A = U \Sigma V^T$  - esempio:



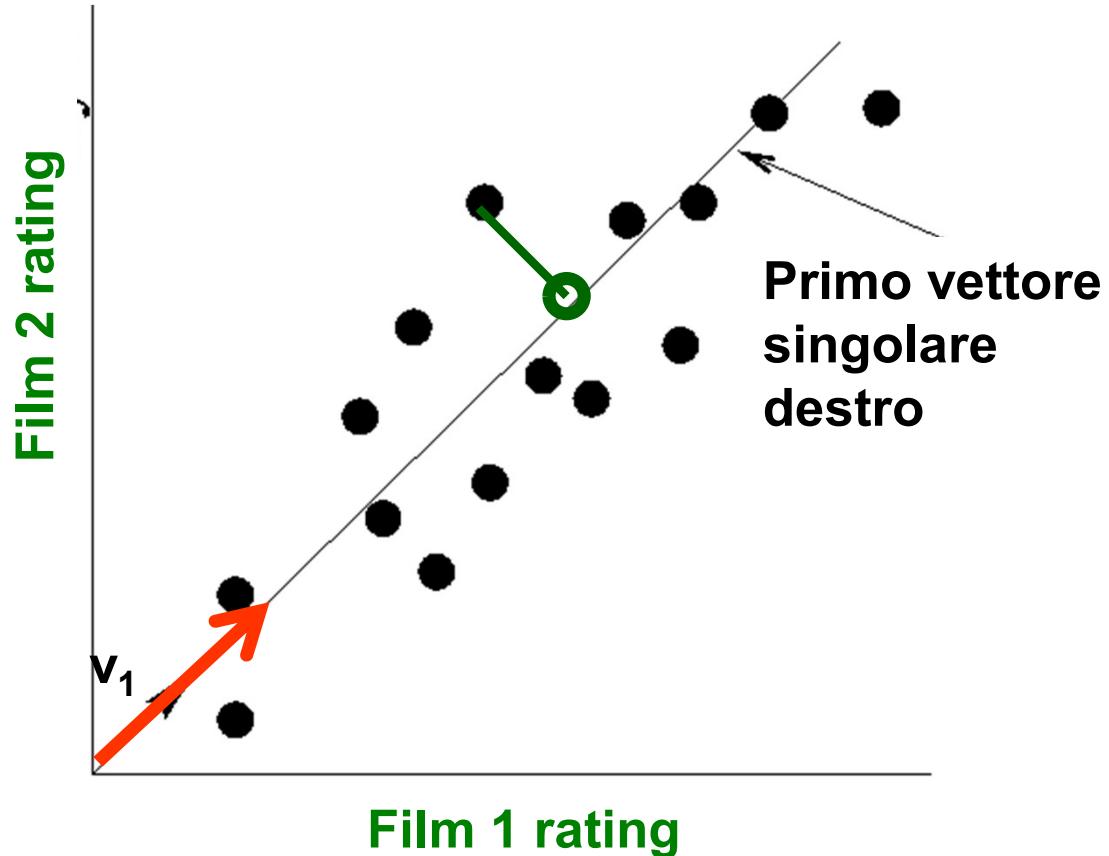
# Dimensionality Reduction con SVD

# SVD – Interpretazione #1

‘**film**’, ‘**utenti**’ and ‘**concetti**’:

- $U$ : matrice di similarità utenti-concetti
- $V$ : matrice di similarità film-concetti
- $\Sigma$ : ogni elemento diagonale misura la “forza” di ogni concetto

# SVD – Dimensionality Reduction



- Invece di usare due coordinate  $(x, y)$  per descrivere la posizione di un punto, usiamo solo una coordinata ( $z$ )
- La posizione di un punto è la sua posizione lungo  $\mathbf{V}_1$
- **Come scegliamo  $v_1$  ? Minimizzare l'errore di ricostruzione**

La **Singular Value Decomposition (SVD)** è una tecnica fondamentale per la **riduzione della dimensionalità**. L'idea centrale è di rappresentare i dati in uno spazio di dimensioni inferiori mantenendo la maggior parte della variabilità o "energia" presente nei dati originali.

Nel contesto del tuo esempio, passare dall'uso di due coordinate  $(x, y)$  a una singola coordinata (**M**) per descrivere la posizione di un punto implica una **proiezione dei dati da uno spazio bidimensionale a uno spazio unidimensionale** (una linea). La domanda cruciale è **come scegliere questa linea (o in generale, questo sottospazio di dimensioni inferiori) in modo ottimale**.

La SVD fornisce un modo sistematico per fare questa scelta basato sul principio di **minimizzare l'errore di ricostruzione**.

Ecco come funziona il concetto di riduzione della dimensionalità con la SVD:

- **Decomposizione:** Partiamo dalla matrice di dati originale  $A$ . Come sappiamo, la SVD la decomponete in  $A = U\Sigma V^T$ .
- **Valori Singolari e Importanza delle Dimensioni:** La matrice diagonale  $\Sigma$  contiene i valori singolari  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0$ , che rappresentano la "forza" o l'importanza di ciascuna dimensione latente o "concetto". I valori singolari più grandi corrispondono alle direzioni nello spazio dei dati originali che hanno la maggiore varianza.
- **Riduzione della Dimensionalità tramite Troncamento:** Per ridurre la dimensionalità a  $k$  dimensioni (dove  $k < r$ ), approssimiamo la matrice  $A$  utilizzando solo i primi  $k$  valori singolari più grandi e i loro corrispondenti vettori singolari sinistri (in  $U$ ) e destri (in  $V$ ). Questa approssimazione di rango  $k$  è data da  $A_k = U_k \Sigma_k V_k^T$ , dove  $U_k$  è composta dalle prime  $k$  colonne di  $U$ ,  $\Sigma_k$  è una matrice diagonale  $k \times k$  contenente i primi  $k$  valori singolari, e  $V_k$  è composta dalle prime  $k$  colonne di  $V$  (quindi  $V_k^T$  ha le prime  $k$  righe di  $V^T$ ).
- **Scelta delle Nuove Coordinate:** Le colonne di  $U_k$  (o le righe di  $V_k^T$  se consideriamo la trasformazione delle colonne di  $A$ ) definiscono le direzioni principali (le nuove "coordinate") nello spazio a  $k$  dimensioni. Nel tuo esempio di passare da due a una coordinata, sceglieremmo la prima colonna di  $U$  (corrispondente al valore singolare  $\sigma_1$ ) come la direzione principale  $x'$  lungo la quale proiettare i punti. La posizione di un punto nello spazio a una dimensione sarà data dalla sua proiezione su questo primo vettore singolare sinistro, scalata dal primo valore singolare.
- **Minimizzazione dell'Errore di Ricostruzione:** La scelta di mantenere le  $k$  dimensioni corrispondenti ai  $k$  valori singolari più grandi è ottimale nel senso che minimizza l'errore di ricostruzione della matrice originale  $A$  dalla sua approssimazione  $A_k$ . L'errore di ricostruzione può essere misurato, ad esempio, dalla norma di Frobenius

della differenza tra  $A$  e  $A_k$ . Il teorema fondamentale della SVD a basso rango afferma che  $A_k$  è la migliore approssimazione di rango  $k$  di  $A$ . Ciò significa che nessun'altra matrice di rango  $k$  può approssimare  $A$  con un errore inferiore (misurato dalla norma di Frobenius).

Nel tuo esempio specifico di riduzione da due a una dimensione:

- La SVD della matrice di dati (dove ogni riga è un punto con due coordinate) ci darebbe due valori singolari,  $\sigma_1 \geq \sigma_2$ , e due coppie di vettori singolari  $u_1, v_1$  e  $u_2, v_2$ .
- Per ridurre a una dimensione, manterremmo solo il primo valore singolare  $\sigma_1$  e i vettori singolari corrispondenti  $u_1$  e  $v_1$ .
- La nuova coordinata  $x'$  per un punto (originariamente rappresentato da una riga in  $A$ ) sarebbe essenzialmente la sua proiezione sul vettore  $v_1$  (o  $u_1$  a seconda della prospettiva, come menzionato nella fonte), pesata da  $\sigma_1$ . La direzione  $v_1$  (o  $u_1$ ) è quella lungo la quale i dati hanno la maggiore variabilità.
- Scegliamo  $\sigma_1$  e i vettori associati perché la rimozione di  $\sigma_2$  (e  $u_2, v_2$ ) comporta il minor errore di ricostruzione possibile quando approssimiamo la matrice originale con una matrice di rango 1.

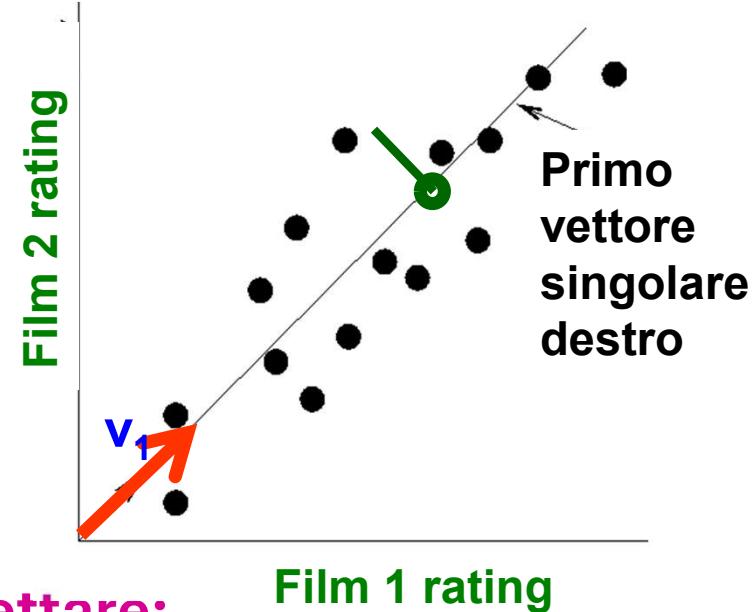
In sintesi, nella riduzione della dimensionalità tramite SVD, la scelta delle nuove coordinate (o delle direzioni principali) è guidata dai valori singolari. Manteniamo le dimensioni corrispondenti ai valori singolari più grandi perché sono quelle che catturano la maggior parte della variabilità dei dati e la loro rimozione comporterebbe il maggiore errore di ricostruzione. L'obiettivo è trovare una rappresentazione a bassa dimensione che sia la "migliore" approssimazione (nel senso di minimizzare l'errore di ricostruzione) dei dati originali.

# SVD – Dimensionality Reduction

- **Goal:** Minimizzare la somma degli errori di ricostruzione:

$$\sum_{i=1}^N \sum_{j=1}^D \|x_{ij} - z_{ij}\|^2$$

- dove  $x_{ij}$  valori “originali” e  $z_{ij}$  sono le “nuove” coordinate



- **SVD restituisce il ‘migliore’ asse dove proiettare:**

- ‘migliore’ = minimizza l’errore di ricostruzione

- **Minimo errore di ricostruzione**

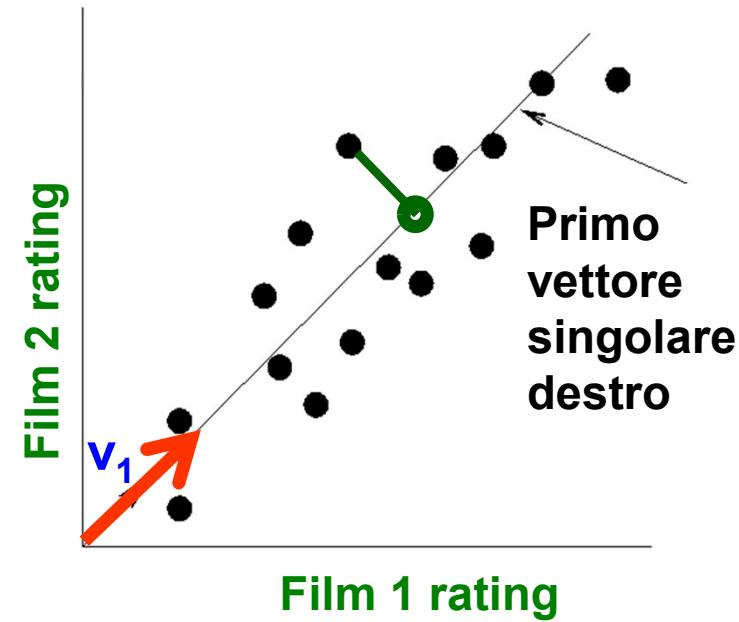
# SVD - Interpretazione

$A = U \Sigma V^T$  - esempio:

- $U \Sigma$ : coordinate dei punti nell'asse di proiezione

1	1	1	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
0	2	0	4	4
0	0	0	5	5
0	1	0	2	2

Proiezione degli utenti sull'asse "Sci-Fi"  $(U \Sigma)^T$ .



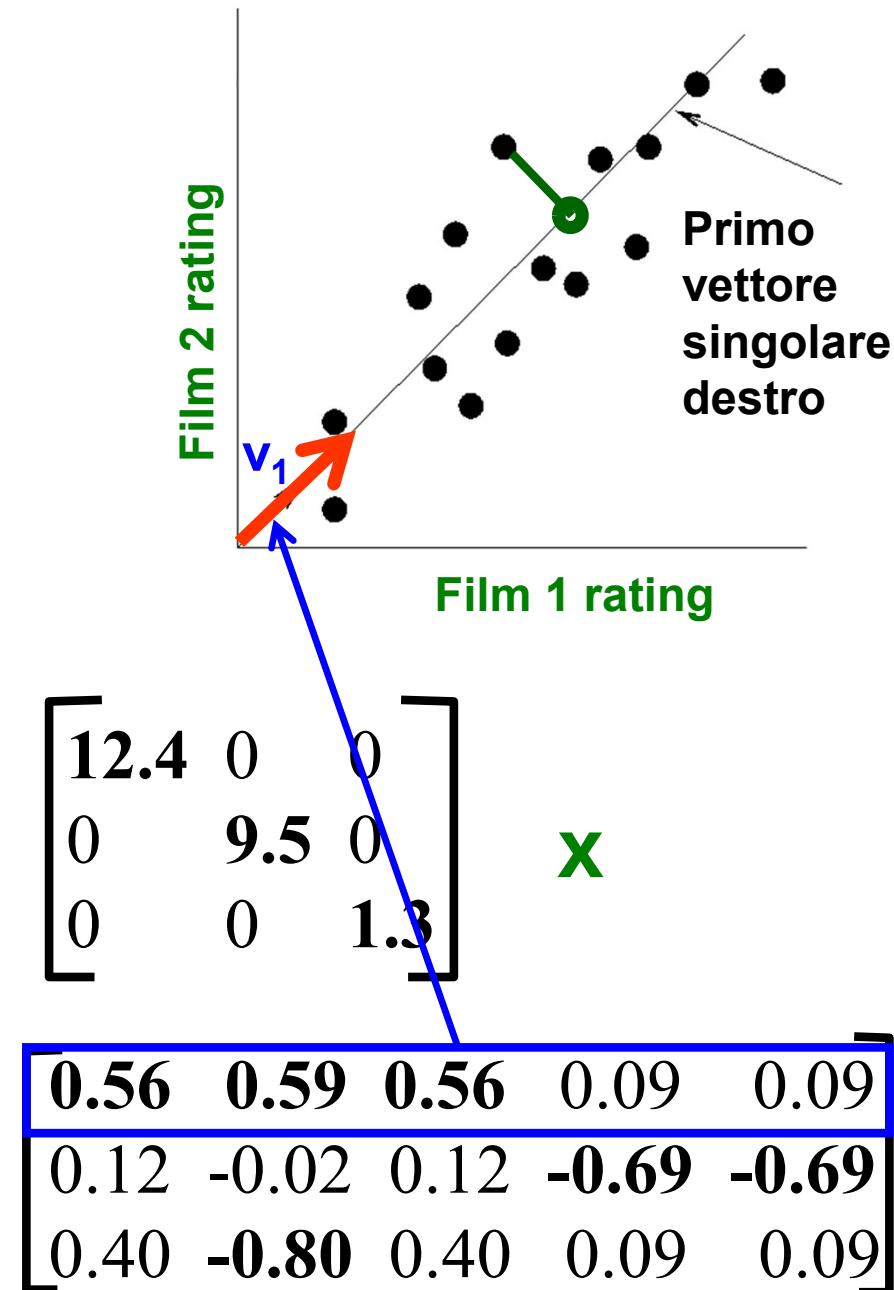
1.61	0.19	-0.01
5.08	0.66	-0.03
6.82	0.85	-0.05
8.43	1.04	-0.06
1.86	-5.60	0.84
0.86	-6.93	-0.87
0.86	-2.75	0.41

# SVD - Interpretazione

## • $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$ - esempio:

- $\mathbf{V}$ : matrice “film-concetti”
- $\mathbf{U}$ : matrice “utenti-concetti”

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$



# SVD - Interpretazione

## Dettagli

- **Q:** Come si reduce la dimensione?

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD - Interpretazione

## Dettagli

- Q: Come si reduce la dimensione?
- A: Settando a zero i valori singolari piu' piccoli

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & \cancel{1.3} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD - Interpretazione #2

## Dettagli

- **Q: Come si reduce la dimensione?**
- **A: Settando a zero i valori singolari piu' piccoli**

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & \cancel{1.3} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD - Interpretazione #2

## Dettagli

- **Q: Come si reduce la dimensione?**
- **A: Settando a zero i valori singolari piu' piccoli**

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

# SVD - Interpretazione #2

## Dettagli

- **Q: Come si reduce la dimensione?**
- **A: Settando a zero i valori singolari piu' piccoli**

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix}$$

# SVD - Interpretazione #2

## Dettagli

- Q: Come si reduce la dimensione?
- A: Settando a zero i valori singolari piu' piccoli

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.92 & 0.95 & 0.92 & 0.01 & 0.01 \\ 2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\ 3.90 & 4.04 & 3.90 & 0.01 & 0.01 \\ 4.82 & 5.00 & 4.82 & 0.03 & 0.03 \\ 0.70 & 0.53 & 0.70 & 4.11 & 4.11 \\ -0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\ 0.32 & 0.23 & 0.32 & 2.01 & 2.01 \end{bmatrix}$$

**Norma di Frobenius:**

$$\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$$

$$\|A-B\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}$$

is "piccolo"

Nel contesto della decomposizione della SVD, la **norma di Frobenius** serve come una **misura fondamentale dell'errore di ricostruzione** quando si approssima la matrice originale  $A$  con una matrice di rango inferiore  $k$ , ottenuta tramite il troncamento della decomposizione SVD.

Ecco i punti chiave sull'importanza della norma di Frobenius nella SVD:

- **Misurazione dell'Errore:** Quando si effettua la **riduzione della dimensionalità**, si scelgono i primi  $k$  valori singolari più grandi e i corrispondenti vettori singolari per costruire un'approssimazione di rango  $k$  della matrice originale  $A$ , indicata come  $A_k$ . La norma di Frobenius ( $\|A - A_k\|_F$ ) quantifica la **differenza complessiva tra la matrice originale  $A$  e la sua approssimazione  $A_k$** . Minore è il valore della norma di Frobenius, migliore è l'approssimazione.
- **Ottimalità della SVD a Basso Rango:** Un **teorema fondamentale** della SVD afferma che l'approssimazione  $A_k$  ottenuta mantenendo i primi  $k$  valori singolari e i loro vettori singolari è la **migliore approssimazione di rango  $k$  della matrice  $A$  nel senso della norma di Frobenius**. Ciò significa che **non esiste un'altra matrice di rango  $k$  che possa approssimare  $A$  con un errore inferiore** misurato dalla norma di Frobenius.
- **Scelta del Numero di Dimensioni  $k$ :** Sebbene la fonte non approfondisca direttamente come scegliere  $k$  usando la norma di Frobenius, il concetto di minimizzare l'errore di ricostruzione (misurato da questa norma) è implicito nella scelta di mantenere i valori singolari più significativi. Un approccio menzionato nella fonte è quello di considerare la **percentuale di "energia" mantenuta** dai primi  $k$  valori singolari (la somma dei loro quadrati rispetto alla somma dei quadrati di tutti i valori singolari). L'obiettivo è mantenere una percentuale elevata (ad esempio, 80-90%), il che indirettamente mira a minimizzare la norma di Frobenius dell'errore.

In sintesi, la norma di Frobenius fornisce una **metrica quantitativa per valutare la qualità dell'approssimazione a basso rango ottenuta tramite la SVD**. La proprietà di ottimalità della SVD rispetto a questa norma garantisce che il troncamento della SVD fornisca la migliore rappresentazione a bassa dimensione possibile in termini di minimizzazione dell'errore di ricostruzione.

# SVD – Best Low Rank Approx.

$$A = U \Sigma V^T$$

B is best approximation of A

$$B = U \Sigma V^T$$

# SVD – Best Low Rank Approx.

- **Teorema:**

Sia  $\mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T$  e  $\mathbf{B} = \mathbf{U} \mathbf{S} \mathbf{V}^T$  dove

$\mathbf{S}$  = diagonale  $rxr$  matrice con  $s_i = \sigma_i$  ( $i=1\dots k$ ) altrimenti  $s_i = 0$   
allora  $\mathbf{B}$  è la migliore  $\text{rank}(\mathbf{B})=k$  approssimazione ad  $\mathbf{A}$

Cosa intendiamo per “migliore”:

- $\mathbf{B}$  soluzione che  $\min_B \|A-B\|_F$  dove  $\text{rank}(B)=k$

$$\begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}_{m \times n} = \begin{pmatrix} u_{11} & \dots & u_{mr} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}_{m \times r} \begin{pmatrix} \sigma_{11} & & & \\ 0 & \ddots & & \\ 0 & & \ddots & \\ \vdots & & & \sigma_{rr} \end{pmatrix}_{r \times r} \begin{pmatrix} v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}_{r \times n}$$

$$\|\mathbf{A}-\mathbf{B}\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}$$

# SVD - Interpretazione #2

**Equivalente:**

‘spectral decomposition’ (decomposizione spettrale) della matrice:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} | & & | \\ u_1 & & u_2 \\ | & & | \end{bmatrix} \times \begin{bmatrix} \sigma_1 & & \\ & \cancel{\sigma_3} & \\ \cancel{\sigma_2} & & \end{bmatrix} \times \begin{bmatrix} v_1 & & \\ & & v_2 \\ & & \end{bmatrix}$$

# SVD - Interpretazione #2

**Equivalente:**

'spectral decomposition'

$$\begin{array}{c} \xleftarrow{\qquad m \quad} \\ \uparrow \quad \downarrow \\ \left[ \begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{array} \right] = \sigma_1 \underbrace{u_1}_{n \times 1} \underbrace{v_1^T}_{1 \times m} + \sigma_2 u_2 v_2^T + \dots \end{array}$$

**k terms**

**Assumiamo:  $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \geq 0$**

**Perché settare  $\sigma_i$  to 0 è la cosa giusta?**

Vettori  $u_i$  e  $v_i$  sono unitari, quindi  $\sigma_i$  scala questi.

Quindi, mettere a zero  $\sigma_i$  piccolo introduce meno errore.

# SVD - Interpretazione #2

**Q: Quanti  $\sigma$ s mantenere?**

**A:** regola empirica:

**mantenere 80-90% dell'‘energia’**  $\sum_i \sigma_i^2$

$$\begin{array}{c} \uparrow \\ \text{n} \\ \downarrow \end{array} \quad \begin{array}{c} \leftarrow \\ \text{m} \\ \rightarrow \end{array} \quad \left[ \begin{array}{ccccc} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{array} \right] = \sigma_1 \ u_1 \ v_1^T + \sigma_2 \ u_2 \ v_2^T + \dots$$

**Assumiamo:**  $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots$

**La somma dei quadrati dei valori singolari dovrebbe essere almeno il 90% della somma dei quadrati di tutti i valori singolari**

## Interpretazione della SVD come somma di matrici di rango 1

Analizziamo la seconda interpretazione della *Singular Value Decomposition* (SVD) presentata nella fonte, che viene definita equivalente alla ‘**spectral decomposition**’.

Questa interpretazione esprime la matrice  $A$  come una **somma di matrici di rango 1**, ciascuna formata dal prodotto esterno di un vettore singolare sinistro ( $u_i$ ) e un vettore singolare destro ( $v_i$ ), scalato dal corrispondente valore singolare ( $\sigma_i$ ).

Formalmente, se la decomposizione SVD di una matrice  $A$  di dimensioni  $n \times m$  è:

$$A = U\Sigma V^T,$$

dove:

- $U = [u_1, u_2, \dots, u_r]$  (con colonne di dimensione  $n \times 1$ ),
  - $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r)$  (di dimensione  $r \times r$ , assumendo che il rango di  $A$  sia  $r$ ),
  - $V = [v_1, v_2, \dots, v_r]$  (con colonne di dimensione  $m \times 1$ ),
- allora  $A$  può essere riscritta come la somma:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_r u_r v_r^T = \sum_{i=1}^r \sigma_i u_i v_i^T.$$

Ogni termine  $\sigma_i u_i v_i^T$  rappresenta una matrice di rango 1 (un prodotto di un vettore colonna per un vettore riga).

### Perché settare $\sigma_i$ a zero è la cosa giusta per la riduzione della dimensionalità?

La fonte spiega che i vettori  $u_i$  e  $v_i$  sono **vettori unitari**, il che significa che la loro “grandezza” è fissata (norma unitaria). Pertanto, il valore singolare  $\sigma_i$  agisce come un **fattore di scala** per la “importanza” o la “contribuzione” della corrispondente matrice di rango 1 ( $u_i v_i^T$ ) nella ricostruzione di  $A$ .

Dato che i valori singolari sono ordinati in modo **decrescente** ( $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \geq 0$ ), i primi termini nella somma (quelli con valori singolari più grandi) contribuiscono maggiormente alla rappresentazione di  $A$ . Settare a zero i valori singolari più piccoli ( $\sigma_i$  per  $i$  elevati) equivale a ignorare

**le componenti meno significative della decomposizione**, introducendo quindi un **errore minore** rispetto a settare a zero valori singolari più grandi. Questo è il principio alla base della riduzione della dimensionalità con la SVD: approssimare  $A$  con una somma troncata dei primi  $k$  termini (corrispondenti ai  $k$  valori singolari più grandi).

## Quanti $\sigma$ da mantenere?

La fonte suggerisce una **regola empirica** per determinare il numero di valori singolari ( $k$ ) da mantenere per una buona approssimazione di  $A$ : **mantenere una quantità di valori singolari tale che la somma dei loro quadrati rappresenti l'80-90% dell'"energia" totale**, dove l'"energia" totale è definita come la somma dei quadrati di tutti i valori singolari.

Matematicamente, questa regola può essere espressa come:

$$\frac{\sum_{i=1}^k \sigma_i^2}{\sum_{i=1}^r \sigma_i^2} \geq 0.8 \text{ (o } 0.9).$$

Dove  $r$  è il rango della matrice  $A$  (o il numero totale di valori singolari non nulli).

Calcolando i valori singolari e i loro quadrati, si può determinare il numero minimo di valori singolari da conservare per raggiungere questa soglia di "energia" mantenuta. Questa è una strategia pratica per bilanciare la riduzione della dimensionalità (riducendo  $k$ ) con la necessità di mantenere una rappresentazione fedele dei dati originali (mantenendo una parte significativa dell'energia).

## Conclusione

In sintesi, questa seconda interpretazione della SVD come somma di matrici di rango 1 pesate dai valori singolari fornisce un'intuizione chiara sul perché il troncamento della SVD, basato sull'ordine decrescente dei valori singolari, è un approccio efficace per la riduzione della dimensionalità. Mantenendo solo i termini con i valori singolari più grandi, si cattura la maggior parte della variabilità dei dati originali con un errore di ricostruzione minimizzato, come discusso anche in precedenza con riferimento alla norma di Frobenius. La regola empirica basata sull'energia fornisce un metodo pratico per decidere quanti componenti principali (corrispondenti ai valori singolari) mantenere.

# Relazione con l'auto-decomposizione

- **SVD :**

- $A = U \Sigma V^T$

- **Eigen-decomposition:**

- $A = X \Lambda X^T$ 
  - A simmetrica
  - U, V, X are ortonormali ( $U^T U = I$ ),
  - $\Lambda$ ,  $\Sigma$  diagonali

- **calcoliamo:**

- $AA^T = U\Sigma V^T (U\Sigma V^T)^T = U\Sigma V^T (V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$
- $A^T A = V \Sigma^T U^T (U\Sigma V^T) = V \Sigma\Sigma^T V^T$

# Relazione con Eigen-decomposition

- **SVD :**

- $A = U \Sigma V^T$

- **Eigen-decomposition:**

- $A = X \Lambda X^T$

- A simmetrica

- U, V, X are ortonormali ( $U^T U = I$ ),

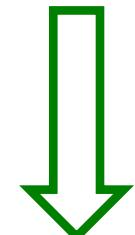
- $\Lambda$ ,  $\Sigma$  diagonali

- **calcoliamo:**

- $AA^T = U\Sigma V^T (U\Sigma V^T)^T = U\Sigma V^T (V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$

- $A^T A = V \Sigma^T U^T (U\Sigma V^T) = V \Sigma \Sigma^T V^T$

Mostra come calcolare  
SVD usando la decomposizione  
Ad autovalori!



$$X \Lambda X^T$$

↓      ↓      ↓

$$\overset{\uparrow}{X} \overset{\uparrow}{\Lambda} \overset{\uparrow}{X^T}$$

# Algortimo per il calcolo dell'SVD

- $\mathbf{A}^T \mathbf{A} = \mathbf{V} \Sigma \Sigma^T \mathbf{V}^T$
- Ne segue che  $\mathbf{V}$  è la matrice degli autovettori e  $\Sigma \Sigma^T$  la matrice degli autovalori
- Usiamo l'algoritmo di power iteration per calcolare la prima autocoppia  $(x, \lambda)$  di  $\mathbf{A}^T \mathbf{A}$ 
  - Sottraiamo l'effetto dell'autocoppia alla matrice  $M' = \mathbf{A}^T \mathbf{A} - \lambda x x^T$
  - Usiamo l'algoritmo di power iteration per calcolare la prima autocoppia di  $M'$ . Iteriamo questo processo per ottenere tutte le autocoppe di  $\mathbf{A}^T \mathbf{A}$ .
- $\mathbf{A} \mathbf{A}^T = \mathbf{U} \Sigma \Sigma^T \mathbf{U}^T$
- Usiamo lo stesso algoritmo per calcolare gli autovettori di  $\mathbf{A} \mathbf{A}^T$

# SVD - Complessità

- **Per calcolare SVD:**
  - $O(nm^2)$  o  $O(n^2m)$  (il minore dei due)
- **MA:**
  - Meno lavoro se vogliamo solo i valori singolari
  - O se vogliamo i primi k valori singolari
  - O se la matrice è sparsa
- **Implementata** in diversi pacchetti
  - LINPACK, Matlab, SPlus, Mathematica, R

# SVD - ricapitolando

- **SVD:  $A = U \Sigma V^T$ : uniche**
  - **U**: similarità utenti-concetti
  - **V**: similarità film-concetti
  - **$\Sigma$**  : forza (importanza) di ogni concetto
- **Dimensionality reduction:**
  - Mantenere i valori singolari che hanno (80-90% dell'‘energia’)
  - SVD: correlazioni lineari

# Relazione tra la Singular Value Decomposition (SVD) e l'Eigen-decomposition

Analizziamo la relazione tra la **Singular Value Decomposition (SVD)** e l'**Eigen-decomposition** basandoci sulle formule fornite.

## SVD

La SVD di una matrice  $A$  è data da:

$$A = U\Sigma V^T$$

dove:

- $U$  è una matrice ortonormale le cui colonne sono i **vettori singolari sinistri**. L'ortonormalità implica che  $U^T U = I$ , dove  $I$  è la matrice identità.
- $\Sigma$  è una matrice diagonale i cui elementi diagonali non negativi sono i **valori singolari** ( $\sigma_i$ ) disposti in ordine decrescente.
- $V$  è una matrice ortonormale le cui colonne sono i **vettori singolari destri**. L'ortonormalità implica che  $V^T V = I$ .

## Eigen-decomposition

L'Eigen-decomposition di una matrice  $A$  è data da:

$$A = X\Lambda X^T$$

Questa decomposizione è tipicamente definita per **matrici simmetriche**,  
dove  $A = A^T$ . In questo caso:

- $X$  è una matrice ortonormale le cui colonne sono gli autovettori della matrice  $A$ . L'ortonormalità implica che  $X^T X = I$ .
- $\Lambda$  è una matrice diagonale i cui elementi diagonali sono gli autovalori ( $\lambda_i$ ) corrispondenti agli autovettori in  $X$ .

## Calcolo di $AA^T$ e $A^T A$ usando la SVD

Consideriamo i prodotti  $AA^T$  e  $A^T A$  utilizzando la decomposizione SVD di  $A$ :

1.  $\boxed{AA^T}$ :

$$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = (U\Sigma V^T)(V^T \Sigma^T (V^T)^T) = (U\Sigma V^T)(V^T \Sigma V) = U\Sigma(V^T V) \Sigma^T = U\Sigma^2 V^T$$

Ricordando che  $(BC)^T = C^T B^T$ , abbiamo  $(U\Sigma V^T)^T = (V^T)^T \Sigma^T U^T = V\Sigma^T U^T$ . Poiché  $\Sigma$  è una matrice diagonale,  $\Sigma^T = \Sigma$ . Quindi:

$$AA^T = (U\Sigma V^T)(V\Sigma U^T) = U\Sigma(V^T V) \Sigma U^T$$

$\downarrow I \Rightarrow V \text{ ortonormale!}$

Dato che  $V$  è ortonormale,  $V^T V = I$ :

$$AA^T = U\Sigma I \Sigma U^T = \boxed{U\Sigma^2 U^T}$$

dove  $\boxed{\Sigma^2}$  è una matrice diagonale con elementi  $(\sigma_i)^2$ .

2.  $\boxed{A^T A}$ :

$$A^T A = (U\Sigma V^T)^T (U\Sigma V^T)$$

Utilizzando il risultato precedente per la trasposta di  $A$ :

$$A^T A = (V\Sigma U^T)(U\Sigma V^T) = V\Sigma(U^T U) \Sigma V^T$$

Dato che  $U$  è ortonormale,  $U^T U = I$ :

$$A^T A = V\Sigma I \Sigma V^T = V\Sigma^2 V^T$$

dove  $\Sigma^2$  è ancora una matrice diagonale con elementi  $(\sigma_i)^2$ .

## Relazione con l'Eigen-decomposition

Confrontando questi risultati con la forma dell'Eigen-decomposition ( $A = X\Lambda X^T$ ), possiamo osservare le seguenti relazioni:

- La decomposizione  $AA^T = U\Sigma^2 U^T$  mostra che:
- Gli autovettori di  $AA^T$  sono le colonne di  $U$ , ovvero i **vettori singolari sinistri** di  $A$ .
  - Gli autovalori di  $AA^T$  sono gli elementi diagonali di  $\Sigma^2$ , ovvero i **quadrati dei valori singolari** di  $A$  ( $\sigma_i^2$ ).
- La decomposizione  $A^T A = V\Sigma^2 V^T$  mostra che:
- Gli autovettori di  $A^T A$  sono le colonne di  $V$ , ovvero i **vettori singolari destri** di  $A$ .
  - Gli autovalori di  $A^T A$  sono anche gli elementi diagonali di  $\Sigma^2$ , ovvero i **quadrati dei valori singolari** di  $A$  ( $\sigma_i^2$ ).

# Algoritmo per il calcolo della Singular Value Decomposition (SVD)

Spieghiamo l'algoritmo per il calcolo della SVD basandoci sulle informazioni fornite.

## Algoritmo per il calcolo dell'SVD

La fonte illustra un approccio per calcolare la SVD sfruttando la relazione con l'**auto-decomposizione** delle matrici  $A^T A$  e  $AA^T$ .

### 1. Calcolo di $V$ e $\Sigma$ :

Dalla relazione:

$$A^T A = V \Sigma^T U^T U \Sigma V^T = V \Sigma^2 V^T,$$

corretto da  $\Sigma \Sigma^T$ , poiché  $\Sigma$  è diagonale e reale ( $\Sigma^T = \Sigma$ ), si deduce che:

- La matrice  $V$  dei **vettori singolari destri** è la matrice degli **autovettori** di  $A^T A$ .

- La matrice  $\Sigma^2$  contiene gli **autovalori** di  $A^T A$  sulla sua diagonale (che sono i quadrati dei valori singolari di  $A$ ,  $\sigma_i^2$ ).

Per calcolare la prima autocoppia (autovettore e autovalore) di  $A^T A$ , si utilizza l'**algoritmo di power iteration**. Questo algoritmo iterativo, applicato a una matrice, converge all'autovettore dominante (corrispondente all'autovalore di modulo maggiore).

Una volta trovata la prima autocoppia  $(\mathbf{x}_1, \lambda_1)$  di  $A^T A$  (che corrisponde al primo vettore singolare destro  $v_1$  e al primo autovalore  $\sigma_1^2$ ), si sottrae l'effetto di questa autocoppia dalla matrice  $A^T A$  per ottenere una nuova matrice:

$$M' = A^T A - \lambda_1 \mathbf{x}_1 \mathbf{x}_1^T.$$

Questa tecnica è nota come **deflazione**.

Si applica nuovamente l'algoritmo di power iteration a  $M'$  per calcolare la sua prima autocoppia, che corrisponderà alla seconda autocoppia di  $A^T A$   $(\mathbf{x}_2, \lambda_2)$  (il secondo vettore singolare destro  $v_2$  e il secondo autovalore  $\sigma_2^2$ ).

Questo processo di iterazione e deflazione viene ripetuto fino ad ottenere tutte le autocoppe desiderate di  $A^T A$ , fornendo così la matrice  $V$  (le cui colonne sono gli autovettori trovati) e i valori  $\sigma_i^2$  (gli autovalori trovati), da cui si ottengono i valori singolari  $\sigma_i = \sqrt{\lambda_i}$  e quindi la matrice  $\Sigma$ .

## 2. Calcolo di $U$ :

Analogamente, dalla relazione:

$$AA^T = U\Sigma V^T(V\Sigma U^T)^T = U\Sigma^2 U^T,$$

si deduce che:

- La matrice  $U$  dei **vettori singolari sinistri** è la matrice degli **autovettori** di  $AA^T$ .

-  $\Sigma^2$  contiene i suoi autovalori.

Si applica lo stesso algoritmo di power iteration con la tecnica di deflazione alla matrice  $AA^T$  per calcolare le sue autocoppie, ottenendo così la matrice  $U$  (le cui colonne sono gli autovettori di  $AA^T$ ). I valori singolari  $\sigma_i$  sono gli stessi ottenuti dall'auto-decomposizione di  $A^T A$  (fatta eccezione per eventuali valori singolari nulli extra, a seconda delle dimensioni di  $A$ ).

## Complessità computazionale

- Il calcolo completo della SVD di una matrice  $n \times m$  ha una complessità di  $O(nm^2)$  o  $O(n^2m)$ , a seconda di quale tra  $m$  e  $n$  sia minore.

- Tuttavia, se si desiderano solo i primi  $k$  valori singolari e i corrispondenti vettori singolari, il costo computazionale può essere significativamente inferiore.

- Anche la sparsità della matrice  $A$  può influenzare positivamente la velocità di calcolo.

- La SVD è implementata in diverse librerie software come LINPACK, Matlab, SPlus, Mathematica e R.

## Ricapitolando la SVD

- La SVD di una matrice  $A$  è data da  $A = U\Sigma V^T$ , dove  $U$  e  $V$  sono matrici ortonormali e  $\Sigma$  è una matrice diagonale con valori singolari non negativi.

-  $U$  può essere interpretata come una matrice che cattura la **similarità tra gli utenti e i concetti latenti** (nell'esempio di sistemi di raccomandazione).

-  $V$  può essere interpretata come una matrice che cattura la **similarità tra gli elementi (film) e i concetti latenti**.

-  $\Sigma$  indica la **forza o l'importanza di ciascun concetto latente**. Valori singolari più grandi corrispondono a concetti che spiegano una maggiore variabilità nei dati.

## Riduzione della dimensionalità con la SVD

- La riduzione della dimensionalità si ottiene **mantenendo solo i valori singolari più grandi** (e le corrispondenti colonne in  $U$  e  $V$ ) che catturano una frazione significativa dell'**energia** totale (solitamente l'80-90%, calcolata come la somma dei quadrati dei valori singolari).
- La SVD è efficace per catturare **correlazioni lineari** nei dati.

## In sintesi

La SVD fornisce una generalizzazione dell'Eigen-decomposition a matrici non necessariamente quadrate o simmetriche.

- Per una matrice **simmetrica**  $A$ , la sua Eigen-decomposition può essere vista come un caso speciale della SVD dove i vettori singolari sinistri e destri sono gli stessi (o al più differiscono per un segno),  $U = V = X$ , e i valori singolari sono i valori assoluti degli autovalori,  $|\lambda_i| = \sigma_i$ .

- In generale, per una matrice  $A$  qualsiasi (non necessariamente simmetrica):

- I vettori singolari sinistri  $U$  sono gli autovettori della matrice simmetrica  $AA^T$ .

- I vettori singolari destri  $V$  sono gli autovettori della matrice simmetrica  $A^TA$ .

- I valori singolari  $\sigma_i$  sono le radici quadrate degli autovalori (non negativi) sia di  $AA^T$  che di  $A^TA$ .

Questa connessione è fondamentale perché permette di calcolare la SVD attraverso la ricerca degli autovalori e degli autovettori delle matrici simmetriche  $AA^T$  e  $A^TA$ . Tuttavia, in pratica, algoritmi numerici diretti per la SVD sono spesso preferiti per ragioni di stabilità numerica.

L'algoritmo descritto si basa sull'idea di trovare gli autovettori e gli autovalori delle matrici  $A^TA$  e  $AA^T$  tramite l'algoritmo di power iteration (e la deflazione per trovare più autocoppie), che corrispondono rispettivamente ai vettori singolari destri e sinistri di  $A$ , e i cui autovalori sono i quadrati dei valori singolari di  $A$ . Questo permette di calcolare la decomposizione SVD.

# Esempio d'uso di SVD e conclusioni

# Case study: come fare delle query?

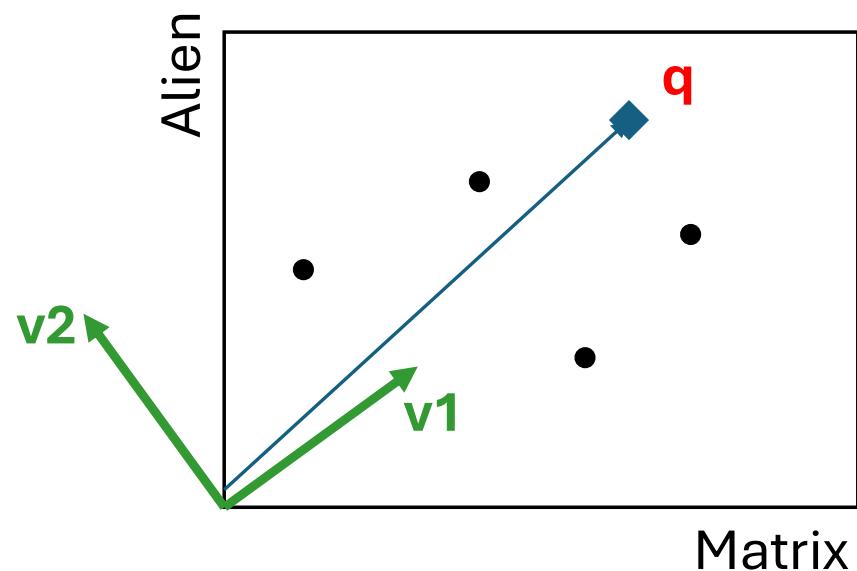
- Q: trovare gli utenti a cui piace ‘Matrix’
- A: Map la query nello spazio dei concetti come?

$$\begin{array}{c}
 \text{↑} \\
 \text{SciFi} \\
 \downarrow \\
 \text{↑} \\
 \text{Romance} \\
 \downarrow
 \end{array}
 \begin{bmatrix}
 \text{Matrix} & 1 & 1 & 1 & 0 & 0 \\
 \text{Alien} & 3 & 3 & 3 & 0 & 0 \\
 \text{Star Wars} & 4 & 4 & 4 & 0 & 0 \\
 \text{Casablanca} & 5 & 5 & 5 & 0 & 0 \\
 \text{Titanic} & 0 & 2 & 0 & 4 & 4 \\
 & 0 & 0 & 0 & 5 & 5 \\
 & 0 & 1 & 0 & 2 & 2
 \end{bmatrix}
 = \begin{bmatrix}
 0.13 & 0.02 & -0.01 \\
 0.41 & 0.07 & -0.03 \\
 0.55 & 0.09 & -0.04 \\
 0.68 & 0.11 & -0.05 \\
 0.15 & -0.59 & 0.65 \\
 0.07 & -0.73 & -0.67 \\
 0.07 & -0.29 & 0.32
 \end{bmatrix}
 \times \begin{bmatrix}
 12.4 & 0 & 0 \\
 0 & 9.5 & 0 \\
 0 & 0 & 1.3
 \end{bmatrix} \times
 \begin{bmatrix}
 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\
 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\
 0.40 & -0.80 & 0.40 & 0.09 & 0.09
 \end{bmatrix}$$

# Case study: come fare delle query?

- Q: trovare gli utenti a cui piace ‘Matrix’
- A: Map la query nello spazio dei concetti come?

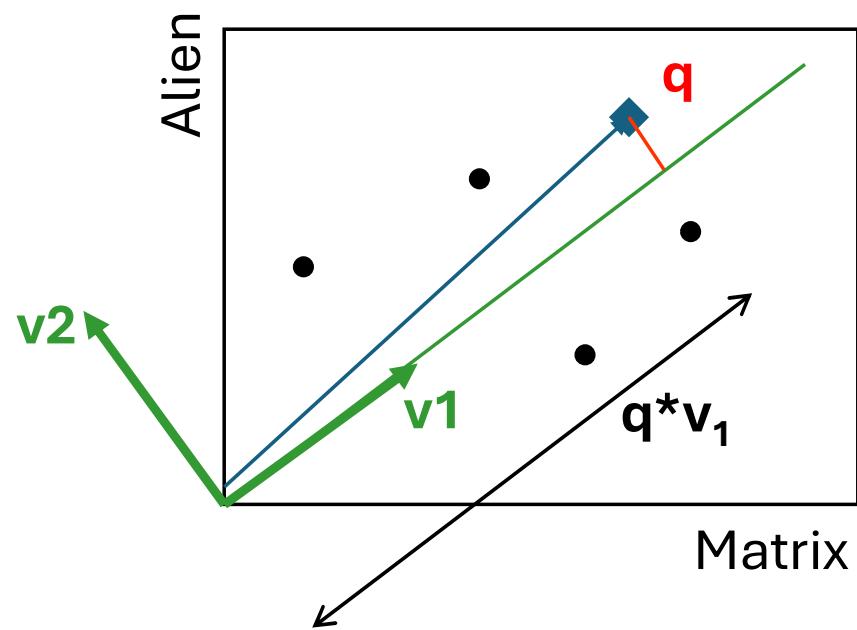
$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \end{bmatrix} \quad \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



# Case study: come fare delle query?

- Q: trovare gli utenti a cui piace ‘Matrix’
- A: Map la query nello spazio dei concetti come?

$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \end{bmatrix} \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



# Case study: come fare delle query?

In modo compatto abbiamo:

$$q_{\text{concept}} = q \mathbf{V}$$

Ese.:

$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \end{bmatrix} \times \begin{bmatrix} 5 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = \begin{bmatrix} 2.8 & 0.6 \end{bmatrix}$$

Similarità film-concetti  
( $\mathbf{V}$ )

Concetto-SciFI

$$\mathbf{q} = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \\ 5 \quad 0 \quad 0 \quad 0 \quad 0 \end{bmatrix}$$

$$\begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix} \times \begin{bmatrix} 2.8 & 0.6 \end{bmatrix}$$

$$\mathbf{q} = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Star Wars} \\ \text{Casablanca} \\ \text{Titanic} \\ 1,64 \quad 1,64 \dots 1,64 \quad 0.16 \quad 0.16 \end{bmatrix}$$

# Case study: come fare delle query?

- Come viene mappato  $d$  che ha valutato ('Alien', 'Star Wars')?  
 $d_{concept} = d V$

Es.:

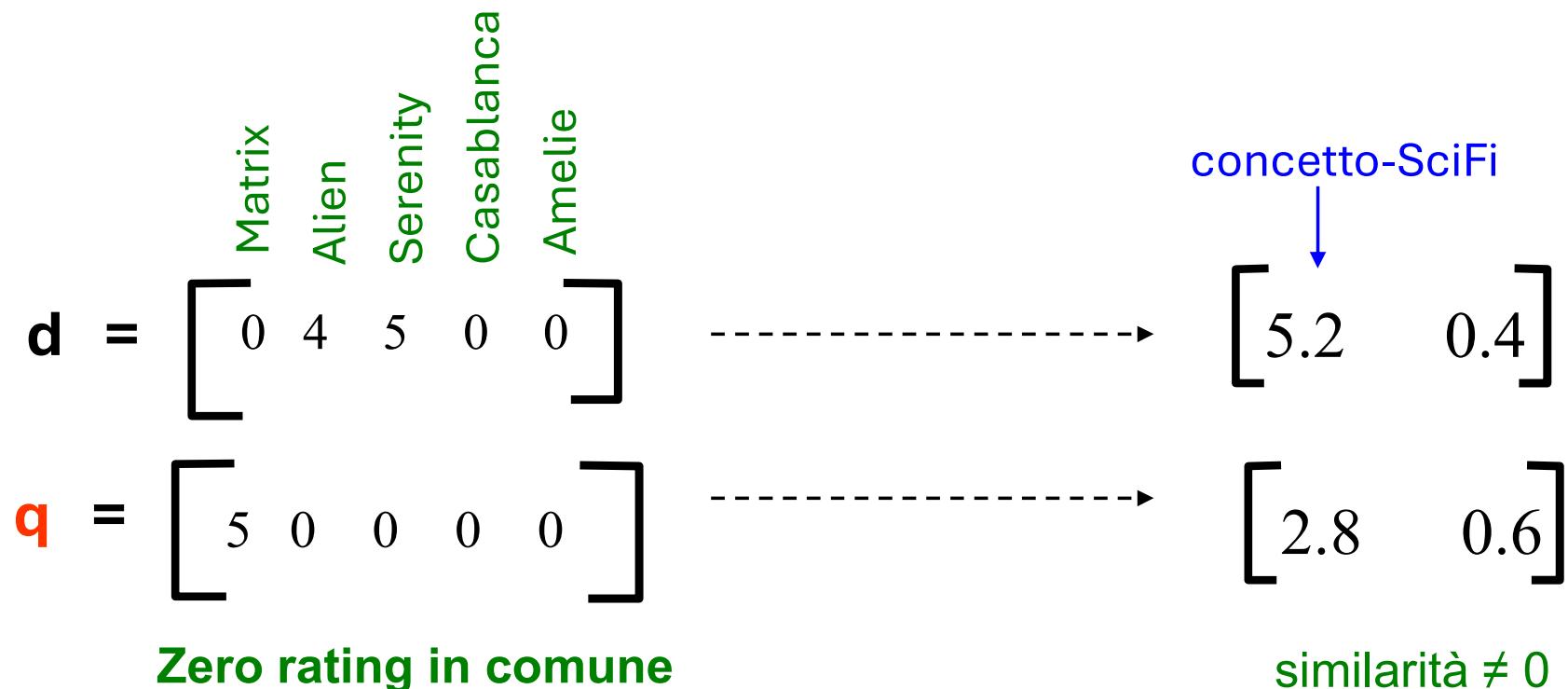
$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ 4 \\ 5 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = \begin{bmatrix} 5.2 \\ 0.4 \end{bmatrix}$$

Concetto-SciFI

Similarità film-concetti  
(V)

# Case study: come fare delle query?

- **Osservazione:** l'utente  $d$  che ha valutato ('Alien', 'Star Wars') sarà **simile** all'utente  $q$  che ha valutato ('Matrix'), sebbene  $d$  e  $q$  non hanno rating in comune!



# SVD: limiti e problemi

+ **Approssimazione low-rank ottimale**

norma di Frobenius

- **Difficile da interpretare:**

- Un vettore singolare specifica una combinazione lineare di colonne e righe di input

- **Mancanza di matrici sparse:**

- I vettori singolari sono **densi!**

$$\begin{matrix} \bullet & \bullet \\ \bullet & \bullet \\ \bullet & \\ \bullet & \bullet \end{matrix} = \begin{matrix} U \\ \Sigma \\ V^T \end{matrix}$$

# Case Study: Uso della SVD per Query in Sistemi di Raccomandazione

Il case study presentato riguarda l'uso della **Scomposizione ai Valori Singolari (SVD)** per risolvere problemi di query in sistemi di raccomandazione basati su matrici di valutazioni utente-film. Analizziamo il concetto passo dopo passo.

## — 1. Contesto del problema

Il problema è quello di trovare gli utenti a cui piace un certo film, ad esempio "Matrix". Questo è un tipico problema di **recupero delle informazioni o query processing** in sistemi di raccomandazione collaborativi.

- La matrice di input rappresenta le **valutazioni degli utenti per i film**, dove: - Le righe corrispondono agli **utenti**. - Le colonne corrispondono ai **film**. - I valori rappresentano le valutazioni (es. da 1 a 5) date dagli utenti ai film.

- Per migliorare la capacità di fare query e raccomandazioni, la matrice viene trasformata usando la **SVD**, che riduce la dimensionalità della matrice originale proiettandola in uno **spazio latente** (conosciuto anche come spazio dei concetti).

## — 2. Scomposizione ai Valori Singolari (SVD)

La SVD decomponе la matrice delle valutazioni  $R$  in tre matrici:

$$R = U\Sigma V^T$$

Dove:  $U$ : Matrice degli utenti nello spazio latente. -  $\Sigma$ : Matrice diagonale contenente i valori singolari (importanza dei concetti). -  $V^T$ : Matrice dei film nello spazio latente.

Lo scopo della SVD è di **ridurre la dimensionalità** mantenendo le informazioni più importanti. Ad esempio, se la matrice originale ha dimensione  $m \times n$ , possiamo approssimarla con una matrice di rango inferiore  $k$  (dove  $k \ll \min(m, n)$ ).

## — 3. Query nello spazio dei concetti

Quando vogliamo fare una query (es. "trovare gli utenti a cui piace 'Matrix'"), dobbiamo: 1. **Mappare la query nello spazio dei concetti.** 2. **Calcolare la similarità tra la query e gli utenti/film in questo spazio.**

Passo 1: Mappare la query nello spazio dei concetti

Supponiamo che la query sia rappresentata da un vettore  $q$ , che indica il film di interesse. Ad esempio: - Se vogliamo trovare gli utenti a cui piace "Matrix", il vettore  $q$  sarà:

$$q = [5, 0, 0, 0, 0]$$

Dove 5 rappresenta una valutazione massima per "Matrix" e 0 indica nessuna valutazione per gli altri film.

Per mappare  $q$  nello spazio dei concetti, lo moltiplichiamo per la matrice  $V$  (che rappresenta i film nello spazio latente):

$$q_{\text{concept}} = q \cdot V$$

Ad esempio, se  $V$  è:

$$V = \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix}$$

Allora:

$$q_{\text{concept}} = [5, 0, 0, 0, 0] \cdot V = [2.8, 0.6]$$

Questo significa che la query "Matrix" è rappresentata nello spazio dei concetti dal vettore  $[2.8, 0.6]$ .

Passo 2: Calcolare la similarità

Una volta che abbiamo la query nello spazio dei concetti ( $q_{\text{concept}}$ ), possiamo calcolare la sua **similarità** con gli utenti o i film in questo spazio. Ad esempio: - Se un utente  $d$  ha valutato "Alien" e "Star Wars", il suo vettore di valutazioni potrebbe essere:

$$d = [0, 4, 5, 0, 0]$$

- Mappiamo  $d$  nello spazio dei concetti:

$$d_{\text{concept}} = d \cdot V = [5.2, 0.4]$$

Ora, calcoliamo la similarità tra  $q_{\text{concept}}$  e  $d_{\text{concept}}$ . Una misura comune è il **coseno della similarità**:

$$\text{Similarità} = \frac{q_{\text{concept}} \cdot d_{\text{concept}}}{\|q_{\text{concept}}\| \cdot \|d_{\text{concept}}\|}$$

In questo caso:

$$\text{Similarità} = \frac{[2.8, 0.6] \cdot [5.2, 0.4]}{\sqrt{2.8^2 + 0.6^2} \cdot \sqrt{5.2^2 + 0.4^2}}$$

Se il risultato è alto, significa che l'utente  $d$  è simile alla query  $q$ , anche se non hanno rating in comune direttamente.

#### 4. Osservazione importante

Un punto chiave del metodo è che l'SVD permette di scoprire **relazioni latenti** tra utenti e film. Anche se due utenti non hanno valutato gli stessi film, possono essere considerati simili se i loro rating si allineano nello spazio dei concetti.

Ad esempio: - L'utente  $q$  ha valutato solo "Matrix". - L'utente  $d$  ha valutato "Alien" e "Star Wars". - Nonostante non abbiano rating in comune, possono essere simili perché entrambi preferiscono ~~film di fantascienza (Sci-Fi)~~.

#### 5. Limiti della SVD

Nonostante i vantaggi, la SVD presenta alcuni limiti:

- **Difficoltà di interpretazione:** I vettori singolari rappresentano combinazioni lineari complesse di utenti e film, rendendo difficile capire cosa rappresentino i concetti latenti.

- **Mancanza di sparsità:** I vettori singolari sono densi, il che può aumentare i costi computazionali quando si lavora con matrici di grandi dimensioni.

- **Approssimazione:** Ridurre la dimensionalità può portare a una perdita di informazioni.

#### Conclusione

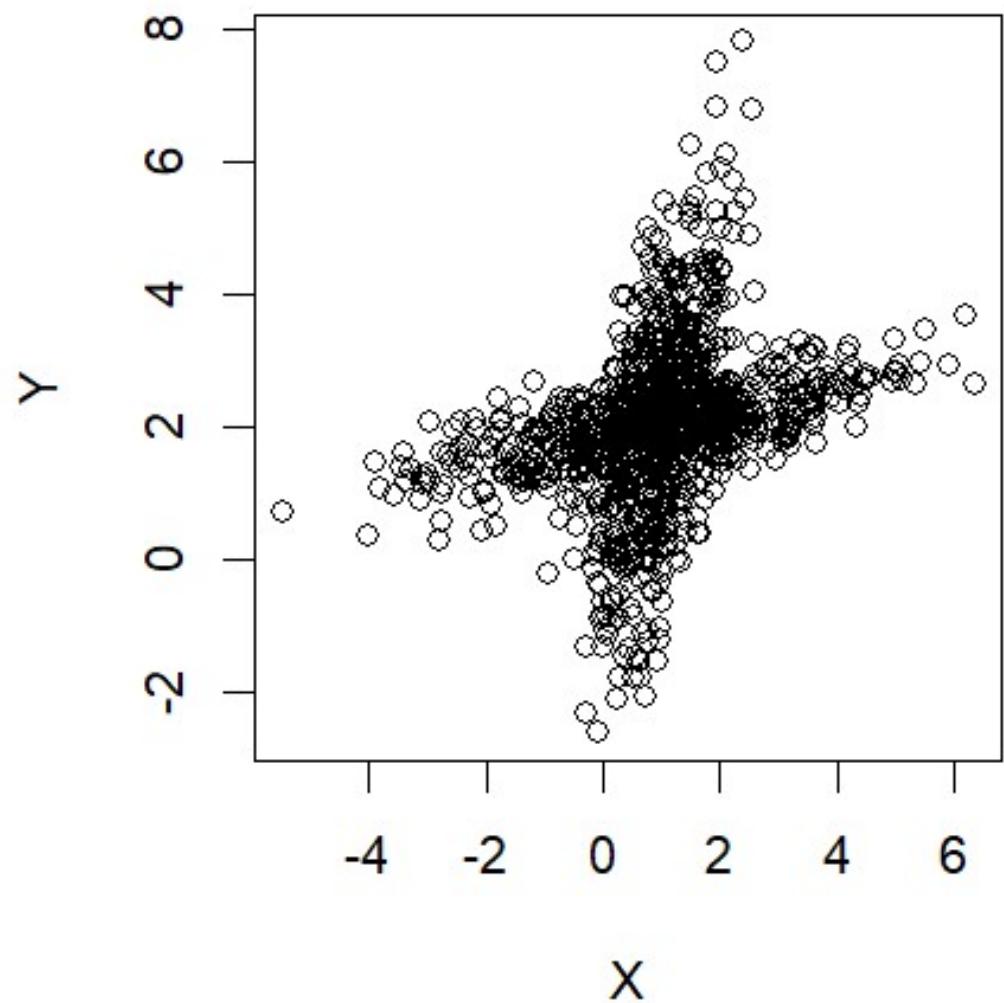
In sintesi: - La SVD è uno strumento potente per mappare query e dati in uno spazio latente, consentendo di scoprire relazioni non evidenti. - Nel tuo case study, la query "trovare gli utenti a cui piace 'Matrix'" viene mappata nello spazio dei concetti tramite la matrice  $V$ . - La similarità tra la query e gli utenti/film viene calcolata nello spazio dei concetti, permettendo di identificare utenti simili anche senza rating in comune.

La query "Matrix" è mappata nello spazio dei concetti come  $q_{\text{concept}} = [2.8, 0.6]$ .

# Decomposizione CUR

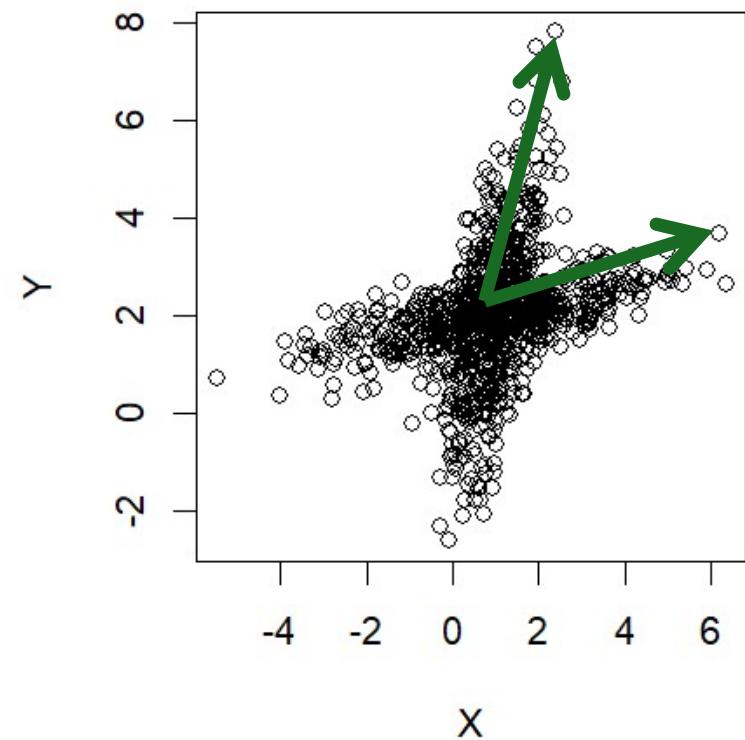
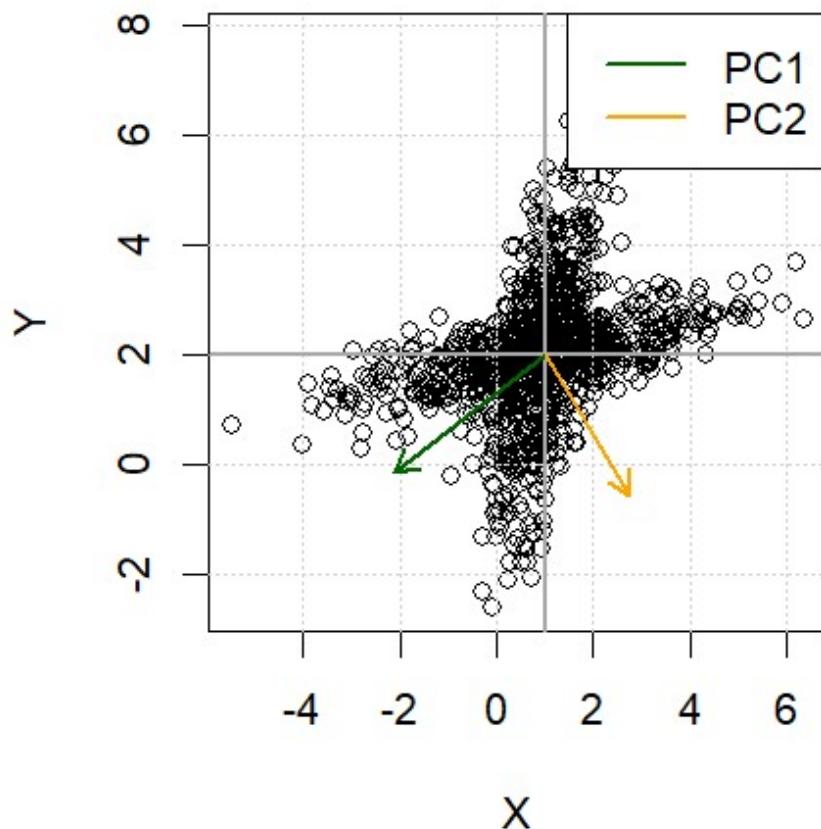
# Semplice motivazione per la decomposizione CUR

- Supponiamo di avere i seguenti dati
  - 1000 vettori a due dimensioni
  - Due gruppi di elementi che vanno su due assi completamente diversi



# Calcoliamo la PCA

- Non catturiamo la direzione dei nostri dati



Norma di Frobenius:  
 $\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}$

# Decomposizione CUR

- Goal: Eprimere  $A$  come il prodotto delle matrici  $C, U, R$   
Tale che  $\|A - C \cdot U \cdot R\|_F$  piccolo
- “Vincoli” su  $C$  e  $R$ :

$$\left( \begin{array}{c|c|c} & & \\ \textcolor{red}{|} & \textcolor{teal}{|} & \textcolor{violet}{|} \\ & A & \\ & \textcolor{teal}{|} & \textcolor{violet}{|} \\ & & \end{array} \right) \approx \left( \begin{array}{c|c|c|c|c} & & & & \\ \textcolor{red}{|} & \textcolor{red}{|} & \textcolor{red}{|} & \textcolor{teal}{|} & \textcolor{violet}{|} \\ & & & & \\ & & & & \end{array} \right) \cdot \left( \begin{array}{c} U \\ \vdots \\ U \end{array} \right) \cdot \left( \begin{array}{c} R \\ \vdots \\ R \end{array} \right)$$

A                    C                    U                    R

Norma di Frobenius:  
 $\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}$

## CUR Decomposition

- Goal: Esprimere  $A$  come il prodotto delle matrici  $C, U, R$

Tale che  $\|A - C \cdot U \cdot R\|_F$  piccolo

- “Vincoli” su  $C$  e  $R$ :

$$\begin{pmatrix} \text{red row} \\ \text{purple row} \\ A \\ \text{blue row} \end{pmatrix} \approx \begin{pmatrix} C \end{pmatrix} \cdot \begin{pmatrix} U \end{pmatrix} \cdot \begin{pmatrix} \text{red row} \\ \text{red row} \\ R \\ \text{purple row} \\ \text{blue row} \end{pmatrix}$$

A                    C                    U                    R

**Pseudo-inversa dell'intersezione  
di  $C$  ed  $R$**

# CUR: Buona approssimazione ad SVD

- **Sia:**

$\mathbf{A}_k$  la “migliore” approssimazione  **$k$  rank**  
ad  $\mathbf{A}$  ( $\mathbf{A}_k$  è l’SVD di  $A$ )

**Teorema** [Drineas et al.]

**CUR** in tempo  $O(m \cdot n)$

- $\|\mathbf{A} - \mathbf{C}\mathbf{U}\mathbf{R}\|_F \leq \|\mathbf{A} - \mathbf{A}_k\|_F + \varepsilon \|\mathbf{A}\|_F$

Con probabilità almeno  $1 - \delta$ , selezionando

- $O(k \log(1/\delta)/\varepsilon^2)$  colonne
- $O(k^2 \log^3(1/\delta)/\varepsilon^6)$  righe

**In pratica:**  
Selezionare  $4k$  col/righe

# CUR: Dettagli

- **Sampling delle colonne (simile per le righe):**

**Input:** matrix  $\mathbf{A} \in \mathbb{R}^{m \times n}$ , sample size  $c$

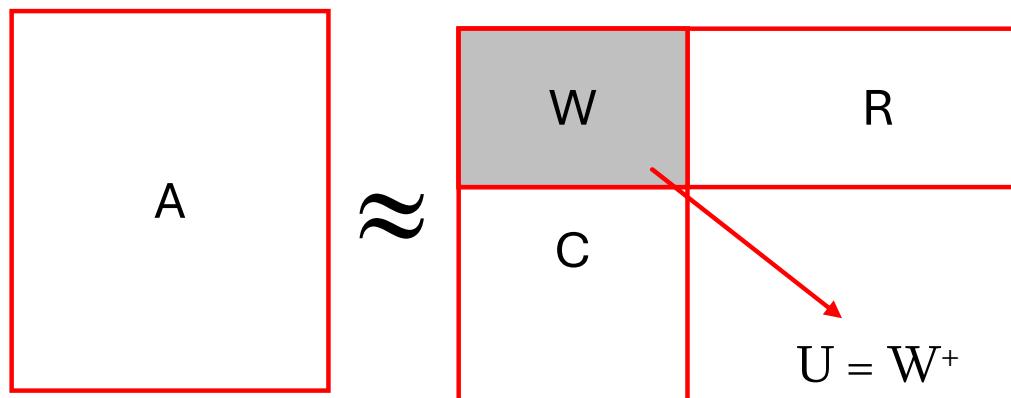
**Output:**  $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for  $x = 1 : n$  [column distribution]
2.  $P(x) = \sum_i \mathbf{A}(i, x)^2 / \sum_{i,j} \mathbf{A}(i, j)^2$
3. for  $i = 1 : c$  [sample columns]
4. Pick  $j \in 1 : n$  based on distribution  $P(x)$
5. Compute  $\mathbf{C}_d(:, i) = \mathbf{A}(:, j) / \sqrt{cP(j)}$

Algoritmo randomizzato, la stessa Colonna potrebbe essere campionata piu' volte

# Come calcolare U

- Sia  $\mathbf{W}$  l’“intersezione” delle colonne e delle righe campionate  $\mathbf{C}$  ed  $\mathbf{R}$ 
  - Calcoliamo l’SVD  $\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}^T$
- Quindi:  $\mathbf{U} = \mathbf{W}^+ = \mathbf{Y} \mathbf{Z}^+ \mathbf{X}^T$ 
  - $\mathbf{Z}^+$ : reciproci dei valori singolari non zero di  $\mathbf{Z}$ :  $Z_{ii}^+ = 1/Z_{ii}$
  - $\mathbf{W}^+$  è la “**pseudoinversa**”



Perché la pseudoinversa funziona?

$\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}$  abbiamo  $\mathbf{W}^{-1} = \mathbf{X}^{-1} \mathbf{Z}^{-1} \mathbf{Y}^{-1}$

Per l'ortonormalità

$\mathbf{X}^{-1} = \mathbf{X}^T$  and  $\mathbf{Y}^{-1} = \mathbf{Y}^T$

Poiché  $\mathbf{Z}$  è diagonale  $\mathbf{Z}^{-1} = 1/Z_{ii}$

Quindi, se  $\mathbf{W}$  è non-singolare, la pseudoinversa è la vera inversa

# CUR: Buona approssimazione ad SVD

$$C = 2 + \epsilon$$

↑  
IF  $\epsilon \rightarrow 0$  then

- **Per esempio:**

- Selezionare  $c = O\left(\frac{k \log k}{\epsilon^2}\right)$  colonne di A con l'algoritmo **ColumnSelect**
- Selezionare  $r = O\left(\frac{k \log k}{\epsilon^2}\right)$  righe di A con l'algoritmo **RowSelect**
- Impostare  $U = W^+$

- **Allora:**

Con probabilità 98%

$$\|A - CUR\|_F \leq (2 + \epsilon) \|A - A_k\|_F$$

**In pratica:**

Selezionare  $4k$  col/righe  
per un approssimazione  
“rank-k”

X

# CUR: Pro & Contro

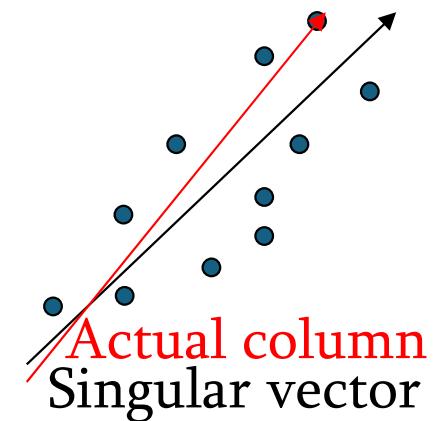
## + Facile da interpretare

- I vettori della base sono esattamente le righe e le colonne della matrice

## + Base sparsa

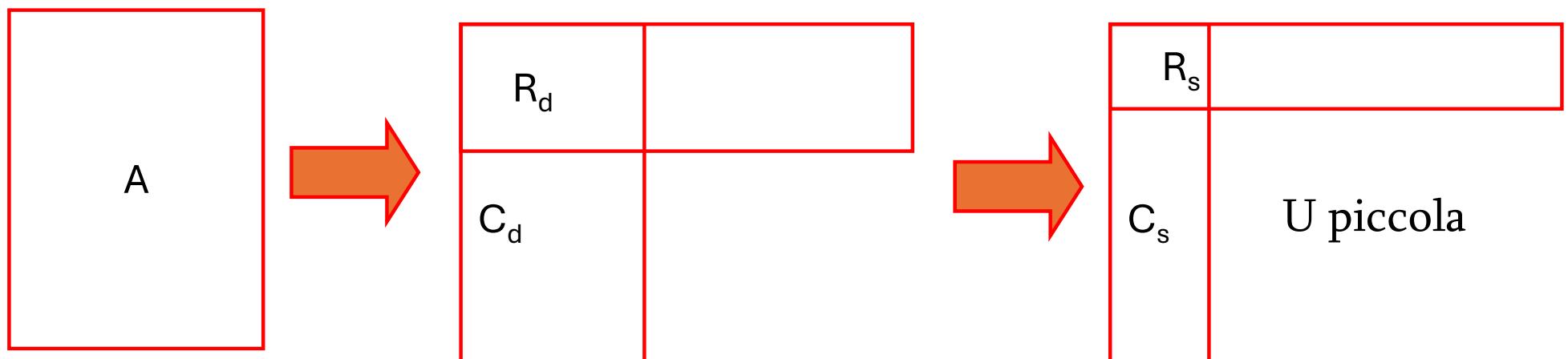
## - Colonne e righe duplicate

- Colonne con norme molto alte verranno selezionate spesso



# Soluzione

- **Se vogliamo sbarazzarci dei duplicati:**
  - Rimuoverli
  - Scalare (moltiplicare) le colonne/righe per la radice quadrata del numero dei duplicati



## SVD vs. CUR

$$\text{SVD: } A = U \Sigma V^T$$

Grande ma sparsa

sparsa e piccola

Grandi e dense

$$\text{CUR: } A = C U R$$

Grande ma sparsa

Densa ma piccola

Grandi ma sparse

# SVD vs. CUR: un esperimento

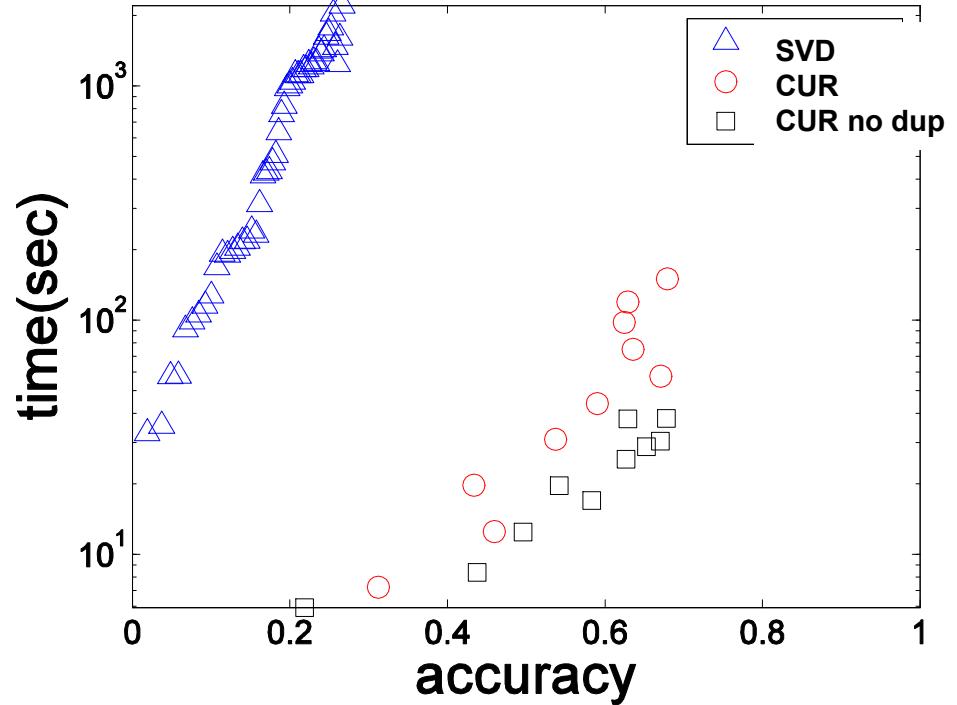
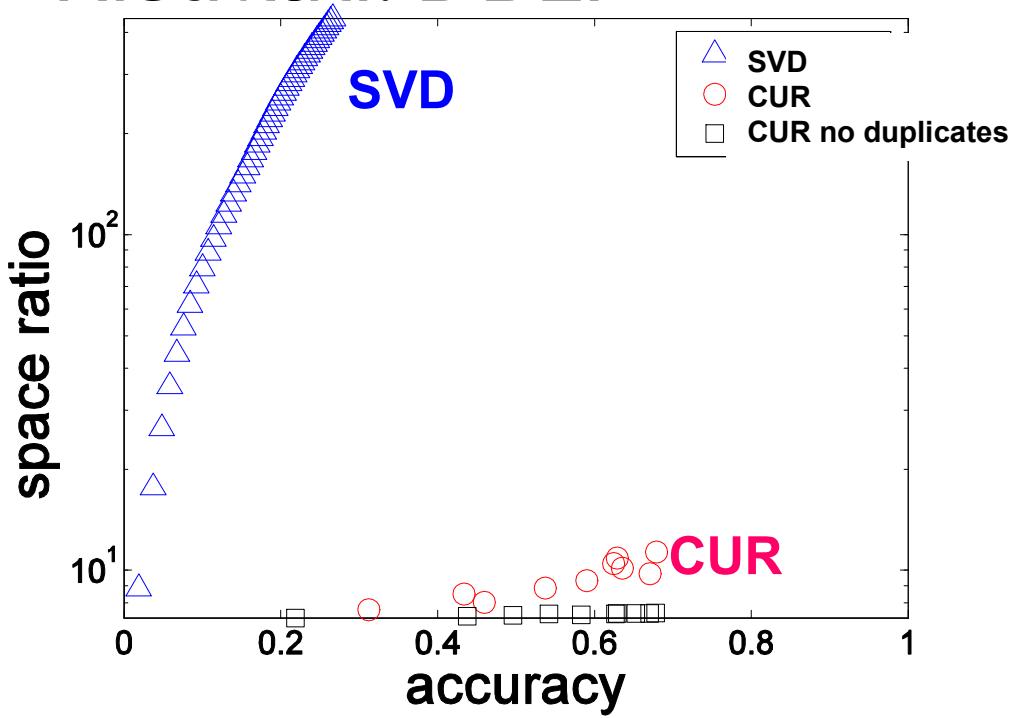
- **DBLP Dati bibliografici**

- Autori-conferenze matrice sparsa molto grande
- $A_{ij}$ : Numero di articoli pubblicati dall'autori  $i$  nella conferenza  $j$
- 428K autori (righe), 3659 conferenze(colonne)
  - **Molto sparsa**

- **Vogliamo ridurre la dimensionalità**

- Quanto tempo ci vuole?
- Qual è l'errore di ricostruzione?
- Quanto spazio ci serve?

# Risultati: DBLP



- **Accuratezza:**
  - 1 – relative sum squared errors
- **Rapporto spazio:**
  - #entry matrice output / #entry matrice input
- **CPU time**

# Approfondimenti: CUR

- Drineas et al., *Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition*, SIAM Journal on Computing, 2006.
- J. Sun, Y. Xie, H. Zhang, C. Faloutsos: *Less is More: Compact Matrix Decomposition for Large Sparse Graphs*, SDM 2007
- *Intra- and interpopulation genotype reconstruction from tagging SNPs*, P. Paschou, M. W. Mahoney, A. Javed, J. R. Kidd, A. J. Pakstis, S. Gu, K. K. Kidd, and P. Drineas, Genome Research, 17(1), 96-107 (2007)
- *Tensor-CUR Decompositions For Tensor-Based Data*, M. W. Mahoney, M. Maggioni, and P. Drineas, Proc. 12-th Annual SIGKDD, 327-336 (2006)

# CUR Decomposition: Spiegazione Dettagliata

## Spiegazione dettagliata della CUR Decomposition

La **decomposizione CUR** è un metodo alternativo alla decomposizione SVD (Singular Value Decomposition) per approssimare una matrice  $A$ . La motivazione principale dietro la CUR decomposition è quella di fornire una rappresentazione più interpretabile e sparsa della matrice originale, utilizzando direttamente alcune colonne e righe della matrice stessa. Vediamo i dettagli passo dopo passo.

### 1. Obiettivo della CUR Decomposition

L'obiettivo è esprimere la matrice  $A$  come il prodotto di tre matrici:

$$A \approx C \cdot U \cdot R$$

dove:

- $C$ : una matrice ottenuta selezionando un sottoinsieme delle colonne di  $A$ .
- $R$ : una matrice ottenuta selezionando un sottoinsieme delle righe di  $A$ .
- $U$ : una matrice che "collega"  $C$  e  $R$ , costruita in modo da minimizzare l'errore di ricostruzione.

L'errore di ricostruzione viene misurato utilizzando la norma di Frobenius, definita come:

$$\|X\|_F = \sqrt{\sum_{i,j} X_{ij}^2}$$

L'obiettivo è quindi minimizzare:

$$\|A - C \cdot U \cdot R\|_F \quad \xrightarrow{\text{Minimiz}} \text{Minimiz}$$

## 2. Vincoli su $C$ e $R$

Le matrici  $C$  e  $R$  devono essere sottomatrici di  $A$ , ovvero devono contenere colonne e righe reali della matrice originale. Questo rende la CUR decomposizione più interpretabile rispetto all'SVD, poiché le colonne e le righe selezionate sono effettivamente presenti nei dati originali.

## 3. Calcolo di $U$

Una volta selezionate le colonne  $C$  e le righe  $R$ , la matrice  $U$  viene calcolata come la **pseudoinversa** dell'intersezione tra  $C$  e  $R$ . L'intersezione è la sottomatrice  $W$  formata dalle righe e colonne comuni a  $C$  e  $R$ .

Il calcolo di  $U$  avviene in due fasi:

1. Calcola la decomposizione SVD di  $W$ :

$$W = X \cdot Z \cdot Y^T$$

dove  $X$  e  $Y$  sono matrici ortogonali, e  $Z$  è una matrice diagonale contenente i valori singolari di  $W$ .

2. Calcola la pseudoinversa di  $W$ :

$$W^+ = Y \cdot Z^+ \cdot X^T$$

dove  $Z^+$  è la matrice diagonale con gli inversi dei valori singolari non nulli di  $Z$ .

Quindi:

$$U = W^+$$

## 4. Selezione delle Colonne e Righe

La selezione delle colonne e righe è cruciale per garantire una buona approssimazione. In pratica:

- Le colonne e le righe vengono campionate in modo **randomizzato**, con probabilità proporzionale alla loro importanza (ad esempio, basandosi sulla norma delle colonne o righe).

- Il numero di colonne e righe selezionate dipende dal parametro  $k$ , che rappresenta il rango desiderato dell'approssimazione.

In particolare: - Selezionando  $O(k \log(k)/\epsilon^2)$  colonne e  $O(k^2 \log^3(1/\delta)/\epsilon^6)$  righe, si garantisce che l'errore di ricostruzione sia piccolo con alta probabilità.

## SAMPLING delle Colonne (righe)

Input = Matrice  $A \in \mathbb{R}^{m \times n}$ , Sample size c  
Output =  $e_d \in \mathbb{R}^{n \times 1}$

For  $x = 1:m$

$$p(x) = \sum_i A_{i,x}^2 / \sum_{i,j} A_{i,j}^2$$

for  $i = 1:c$

Pick  $j \in 1:m$  based on distribution  $p(x)$

$$\text{Compute } C_d(:,i) = A(:,j) / \sqrt{c p(j)}$$

## 5. Teorema di Approssimazione

Esiste un teorema che lega la CUR decomposition all'SVD:

$$\text{Probabilità } 88\% \quad \|A - C \cdot U \cdot R\|_F \leq (2 + \epsilon) \|A - A_k\|_F$$

dove:

- $A_k$  è la migliore approssimazione di rango  $k$  di  $A$  (ottenuta tramite SVD).
- $\epsilon$  è un parametro di errore che può essere reso arbitrariamente piccolo aumentando il numero di colonne e righe selezionate.

Questo teorema mostra che la CUR decomposition è una buona approssimazione dell'SVD, ma con il vantaggio di usare colonne e righe reali della matrice originale.

## 6. Vantaggi e Svantaggi della CUR Decomposition

Vantaggi:

- **Interpretabilità:** Le colonne e le righe selezionate sono direttamente tratte dalla matrice originale, rendendo i risultati più facili da interpretare.
- **Sparsità:** La CUR decomposition preserva la sparsità della matrice originale, il che è utile per matrici molto grandi e sparse.
- **Efficienza computazionale:** Il calcolo della CUR decomposition è generalmente più veloce dell'SVD, specialmente per matrici di grandi dimensioni.

Svantaggi:

- **Duplicati:** È possibile che alcune colonne o righe vengano selezionate più volte, portando a duplicati. Questo può essere risolto rimuovendo i duplicati o scalando le colonne/righe selezionate.
- **Errore di approssimazione:** L'errore di ricostruzione potrebbe essere leggermente maggiore rispetto all'SVD, soprattutto per matrici con una struttura complessa.

## 7. Esempio Pratico: DBLP Dataset

Nell'esperimento descritto, la CUR decomposition è stata applicata a una matrice sparsa di dimensioni enormi (428K autori  $\times$  3659 conferenze). I

risultati mostrano che:

- La CUR decomposition è significativamente più veloce dell'SVD.
- L'errore di ricostruzione è comparabile a quello dell'SVD.
- La CUR decomposition richiede meno spazio di archiviazione rispetto all'SVD, grazie alla sua sparsità.

## 8. Confronto tra SVD e CUR

Caratteristica	SVD	CUR
<b>Interpretabilità</b>	Basso (vettori singolari)	Alto (colonne/ righe reali)
<b>Sparsità</b>	No	Sì
<b>Complessità</b>	Alta	Bassa
<b>Errore di Ricostruzione</b>	Minimo	Leggermente più alto

Table 1: Confronto tra SVD e CUR

## Conclusione

La **CUR decomposition** è un metodo efficace per approssimare matrici grandi e sparse, offrendo un equilibrio tra accuratezza, efficienza e interpretabilità. È particolarmente utile in contesti in cui è importante mantenere la struttura originale dei dati, come nel caso di matrici bibliografiche o reti sociali.

$$A \approx C \cdot U \cdot R$$

# Non Negative Matrix Factorisation

# Nonnegative Matrix Factorisation

- NMF consente una rappresentazione dei dati lineare:

$$A \approx WH$$

- Dove
- $A = [a_{mn}]$
- $W = [w_{mk}]$  tale che  $w_{mk} \geq 0$
- $H = [h_{kn}]$  tale che  $h_{kn} \geq 0$
- Il rango  $k$  della fattorizzazione è scelto in modo tale che  $(m + n)k < mn$ , e il prodotto  $WH$  è una rappresentazione compressa di  $A$ .

# Perche la fattorizzazione a fattori non negativi?

- La nonnegatività induce sparsità.
- Consente una decomposizione in parti

# NMF

- A: Matrice dei dati  $m \times n$ 
  - m features (righe)
  - n osservazioni/esempi/vettori di feature (colonne)
- $\mathbf{a}_i = (a_{1i}, a_{2i}, \dots, a_{mi})^T$ : i-simo vettore di osservazione di feature estratto dagli n
- $\mathbf{a}_i$  vettore colonna in  $\mathbb{R}_+^m$

# W e H

- W : M x K **matrice dizionario:**

- $w_{mk}$  coefficiente della matrice,
- $\mathbf{w}_m$  a dizionario/vettore base con K elementi;

- H : K X N **matrice di attivazione/espansione:**

- $\mathbf{h}_n$  : vettore colonna di coefficienti di attivazione per l'osservazione  $\mathbf{a}_n$ :

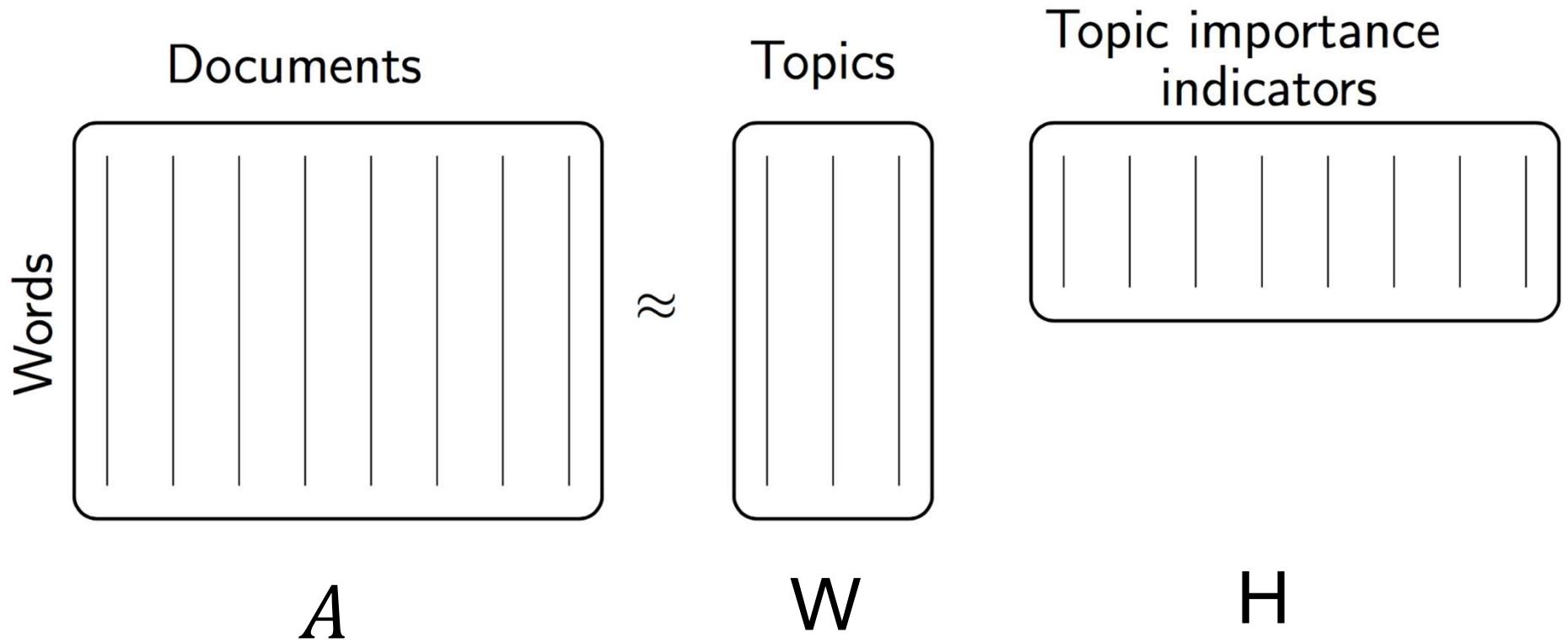
$$a_{fn} \approx \sum_{k=1}^K w_{fk} \times h_{kn}$$

- $\mathbf{h}_k$ : **vettore riga** di coefficienti di attivazione per il vettore base  $\mathbf{w}_f$ .

# Interpretazione di $\mathbf{W}$ e $\mathbf{H}$

- Le colonne di  $\mathbf{W}$  sono i vettori della base, ogni colonna di  $\mathbf{A}$  può essere costruita come combinazione lineare delle  $k$  colonne di  $\mathbf{W}$ .
- Le colonne di  $\mathbf{H}$  danno i pesi (coefficienti) associati ad ogni vettore della base.

- NMF è una decomposizione dei dati non non-supervised, consente di effettuare una analisi delle variabili latenti, che possono essere usati per diversi scopi.
- **Scoperta dei concetti:**
  - assumiamo  $\mathbf{A} = [a_{mn}]$  è una di co-occorrenza matrice termini-documenti:  $a_{mn}$  è la frequenza della  $x_m$  nel documento  $d_n$ ;



# NMF legame con Probabilistic Latent Semantic Analysis (PLSA)

- Modello PLSA (Hofmann, 1999)

$$P(m_f, d_n) = \sum_{k=1}^K P(t_k)P(d_n|t_k)P(m_f|t_k)$$

- I documenti possono essere descritti attraverso alcuni concetti sottostanti  $t_k$ .

- Sia  $w_{fk} = \hat{P}(t_k) \hat{P}(m_f | t_k)$  e  $h_{kn} = \hat{P}(d_n | t_k)$
- Il modello puo' essere riscritto come:  
$$[\hat{P}(m_f d_n)] = [\hat{a}_{fn}] = \mathbf{WH}$$
- Con  $\mathbf{w}_k$  interpretati come concetti che possono spiegare i dati analizzati attraverso i relativi  $\mathbf{h}_k$

# Funzione obiettivo

- NMF Approssimazione  $A \approx WH$  usualmente ottenuta con:

$$\min_{W,H \geq 0} D(A|WH)$$

- Funzione obiettivo Euclidea

$$\sum_{f,n} (a_{fn} - \hat{a}_{fn})^2$$

- Divergenza di Kullback-Leibler (KL)

$$\sum_{f,n} (a_{fn} \log \frac{a_{fn}}{\hat{a}_{fn}} - a_{fn} + \hat{a}_{fn})$$

# Algoritmo iterativo per NMF

$$w_{ia} = w_{ia} \sum_{\mu} \frac{a_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$$
$$w_{ia} = \frac{w_{ia}}{\sum_j w_{ja}}$$

$$h_{a\mu} = h_{a\mu} \sum_i w_{ia} \frac{a_{i\mu}}{(WH)_{i\mu}}$$

L'iterazione di queste regole di aggiornamento convergono ad un massimo locale della funzione obiettivo

$$X = \sum_{i=1}^f \sum_{\mu=1}^n [a_{i\mu} \log(WH)_{i\mu} - (WH)_{i\mu}]$$

# Notebook

[https://colab.research.google.com/drive/15yp-FwlZZeAavVqcR3XnhnvwHYYSMAp\\_?usp=sharing](https://colab.research.google.com/drive/15yp-FwlZZeAavVqcR3XnhnvwHYYSMAp_?usp=sharing)

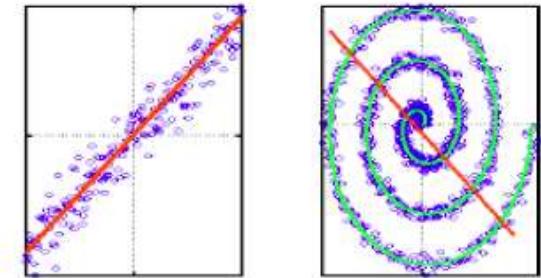
# Pacchetti R per la riduzione della dimensionalità

- **prcomp** e **svd** disponibili come funzioni di base di R
- Package utili:
  - rCUR
  - NMF
  - irlba
  - rARPACK
- Matrixcalc
  - Per il calcolo della Norma di Frobenius

# L'assunzione della linearità

- **SVD è limitata alla proiezione lineare:**

- Lower-dimensional linear projection che preservano la distanza Euclidea

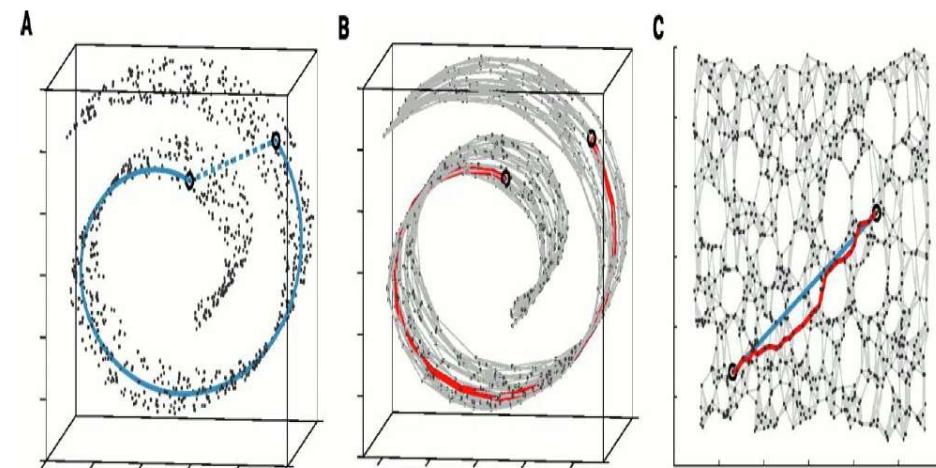


- **Metodi Non-lineari: Isomap**

- I dati cadono in una curva (manifold) a piu' bassa dimensione
  - Usare una distanza congrua per il manifold

- **Come?**

- Grafo di adiacenza
- Distanza geodesica  
è la distanza nel grafo
- SVD/PCA della matrice delle distanze del grafo



# Tecniche piu' recenti

- t-SNE (t-distributed stochastic neighbor embedding )
- <https://lvdmaaten.github.io/tsne/>
- UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction
- <https://umap-learn.readthedocs.io/en/latest/clustering.html>

# Non-Negative Matrix Factorization (NMF): Una Rappresentazione Lineare dei Dati

## 1. Non-Negative Matrix Factorization (NMF): Una Rappresentazione Lineare dei Dati

La **Non-Negative Matrix Factorization (NMF)** è una tecnica di decomposizione matriciale che consente di rappresentare una matrice  $A$  come il prodotto di due matrici più piccole  $W$  e  $H$ :

$$A \approx W \cdot H$$

dove:

- $A$ : Matrice originale di dimensioni  $m \times n$ , dove  $m$  rappresenta le features (righe) e  $n$  le osservazioni o esempi (colonne).
- $W$ : Matrice "dizionario" di dimensioni  $m \times k$ , con elementi non negativi ( $w_{mk} \geq 0$ ). Le colonne di  $W$  sono vettori di base che rappresentano componenti latenti.
- $H$ : Matrice "di attivazione" di dimensioni  $k \times n$ , anch'essa non negativa ( $h_{kn} \geq 0$ ). Le righe di  $H$  contengono i pesi (coefficienti) associati a ciascun vettore di base.

Il rango  $k$  della fattorizzazione è scelto in modo tale che  $(m + n)k < mn$ , garantendo una compressione significativa dei dati. Il prodotto  $W \cdot H$  rappresenta quindi una versione approssimata di  $A$ , mantenendo solo le informazioni essenziali.

### Perché la Fattorizzazione a Fattori Non Negativi?

La nonnegatività impone restrizioni naturali che riflettono meglio la realtà di molti dataset:

- **Sparsità:** La nonnegatività induce sparsità nei dati, rendendo le rappresentazioni più interpretabili.

- **Decomposizione in Parti:** NMF consente di scomporre i dati in componenti additive, utili per scoprire "parti" o "concetti" sottostanti. Ad esempio, in analisi di testi, i vettori di base possono rappresentare argomenti o temi principali.

## 2. Interpretazione di $W$ e $H$ : Vettori di Base e Coefficienti

Le colonne di  $W$  rappresentano i vettori di base, mentre le colonne di  $H$  forniscono i pesi associati a ciascun vettore di base. Questo significa che ogni colonna di  $A$  può essere ricostruita come combinazione lineare delle colonne di  $W$ , ponderate dai coefficienti in  $H$ :

$$a_{fn} \approx \sum_{k=1}^K w_{fk} \cdot h_{kn}$$

### Interpretazione Pratica

- In contesti di elaborazione del linguaggio naturale (NLP),  $W$  può rappresentare "temi" o "argomenti", mentre  $H$  indica quanto ciascun documento è associato a quei temi.

- In immagini,  $W$  può rappresentare parti fondamentali dell'immagine (ad esempio, bordi o texture), mentre  $H$  indica come queste parti si combinano per formare l'immagine completa.

NMF è una tecnica **non supervisionata**, il che significa che non richiede etichette per scoprire strutture latenti nei dati. Questa caratteristica la rende particolarmente utile per scoprire concetti nascosti, come argomenti in documenti o cluster in dati biologici.

### 3. Legame con Probabilistic Latent Semantic Analysis (PLSA)

NMF ha una stretta relazione con il modello PLSA, che è un approccio probabilistico per l'analisi di co-occorrenza termini-documenti. Nel modello PLSA:

$$P(m_f, d_n) = \sum_{k=1}^K P(t_k) P(d_n | t_k) P(m_f | t_k)$$

dove:

- $t_k$  rappresenta i concetti latenti.
- $P(t_k)$  è la probabilità del concetto  $t_k$ .
- $P(d_n | t_k)$  e  $P(m_f | t_k)$  sono rispettivamente le probabilità condizionali del documento  $d_n$  e del termine  $m_f$  dato il concetto  $t_k$ .

Riscrivendo il modello PLSA in termini di NMF:

$$w_{fk} = P(t_k) P(m_f | t_k) : \text{Rappresenta il contributo del concetto } t_k \text{ al termine } m_f.$$

$$h_{kn} = P(d_n | t_k) : \text{Rappresenta il contributo del concetto } t_k \text{ al documento } d_n.$$

Questo legame evidenzia come NMF possa essere interpretata come un metodo per scoprire concetti latenti in dati di co-occorrenza, simile a PLSA ma con un approccio deterministico.

### 4. Funzione Obiettivo e Algoritmi Iterativi

La decomposizione NMF viene ottenuta minimizzando una funzione obiettivo che misura la discrepanza tra  $A$  e  $WH$ . Due metriche comuni sono:

- **Funzione Obiettivo Euclidea:**

$$\min_{W,H \geq 0} \sum_{f,n} (a_{fn} - \hat{a}_{fn})^2$$

- **Divergenza di Kullback-Leibler (KL):**

$$\min_{W,H \geq 0} \sum_{f,n} \left( a_{fn} \log \frac{a_{fn}}{\hat{a}_{fn}} - a_{fn} + \hat{a}_{fn} \right)$$

Queste funzioni obiettivo vengono ottimizzate utilizzando algoritmi iterativi, come le regole di aggiornamento moltiplicative:

$$w_{ia} = w_{ia} \frac{a_{ip}}{\sum_j (W^H)_{jp} H_{aj}}$$

$$h_{ap} = h_{ap} \frac{a_{ip}}{\sum_i w_{ia}}$$

*ITERAZIONI DI QUESTE REGOLE DI AGGIORNAMENTO CONVERGONO AD UN MASSIMO LOCALIZZATO DELLE FUNZIONI OBETTIVO*

$$\chi = \sum_{i=1}^I \sum_{p=1}^P [a_{ip} \log (W^H)_{ip} - (W^H)_{ip}]$$

## 5. Limiti della Linearità e Tecniche Non-Lineari

Mentre NMF e SVD si basano su proiezioni lineari, alcuni dataset presentano strutture non lineari che richiedono tecniche più avanzate:

- **Isomap**: Utilizza un grafo di adiacenza per calcolare distanze geodetiche (lungo il manifold) invece di distanze euclidee.
- **t-SNE**: Riduce la dimensionalità preservando le relazioni locali tra punti, ideale per visualizzare cluster.
- **UMAP**: Combina tecniche di manifold learning con algoritmi efficienti per grandi dataset.

Queste tecniche sono particolarmente utili quando i dati si trovano su un manifold a bassa dimensione, ovvero una curva o superficie intrinsecamente più semplice rispetto allo spazio originale.

## 6. Pacchetti e Strumenti per la Riduzione della Dimensionalità

In ambito pratico, esistono numerosi strumenti per implementare NMF e altre tecniche di riduzione della dimensionalità:

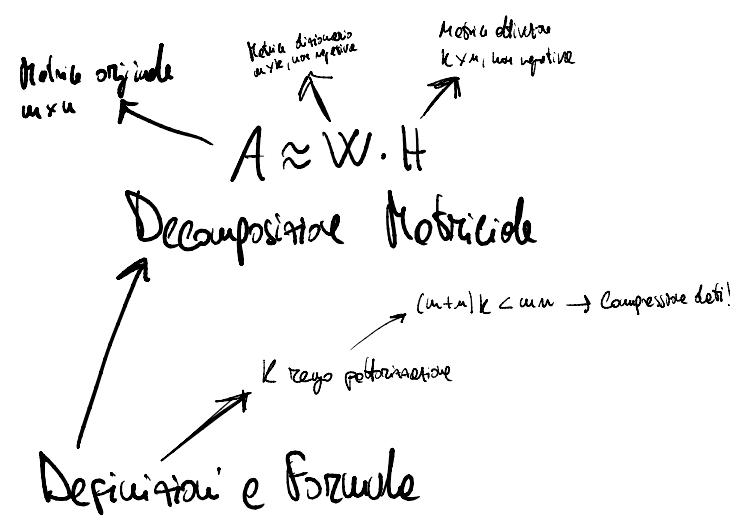
- **In R**:
  - **prcomp** e **svd**: Funzioni base per PCA e SVD.
  - **NMF**: Pacchetto dedicato alla fattorizzazione non negativa.
  - **irlba** e **rARPACK**: Ottimizzati per SVD su matrici sparse.
  - **Matrixcalc**: Calcolo della norma di Frobenius.
- **In Python**:

- Librerie come `scikit-learn` offrono implementazioni efficienti di NMF, t-SNE e UMAP.

## Conclusione

La **Non-Negative Matrix Factorization** è uno strumento potente per la scoperta di strutture latenti nei dati, grazie alla sua capacità di fornire rappresentazioni sparse e interpretabili. Mentre tecniche lineari come NMF e SVD sono ampiamente utilizzate, metodi non lineari come t-SNE e UMAP stanno guadagnando popolarità per gestire complessità più elevate. Comprendere il legame tra NMF e modelli probabilistici come PLSA offre ulteriori spunti per applicazioni in settori come il text mining, la biologia computazionale e l'elaborazione di immagini.

$$A \approx W \cdot H$$



## Non Negative Matrix Factorization

Puntate obiettivi e algoritmi

- Minimizzazione distanteza  $A \approx W \cdot H$
- Funzione obiettivo Euclidea
- Dimensione  $K$
- Algoritmo iterativo
- Regole di aggiornamento multiplicative
- Convergenza punto critico

Legare con PLSA

- Modello probabilistico per co-occurrenza tempi - documenti