Singular value decomposition

A: Generalizing the spectral decomposition

Generalizing the spectral decomposition

For **symmetric** matrices (e.g. covariance matrices), we have seen:

- Results about existence of eigenvalues and eigenvectors
- Eigenvectors form an alternative basis
- Resulting spectral decomposition, used in PCA

What about **arbitrary** matrices $M \in \mathbb{R}^{p \times q}$?

Singular value decomposition (SVD)

Any $p \times q$ matrix ($p \leq q$) has a **singular value decomposition**:

$$M = \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_p \\ \downarrow & & \downarrow \end{pmatrix}}_{p \times p \text{ matrix } U} \underbrace{\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{pmatrix}}_{p \times p \text{ matrix } \Lambda} \underbrace{\begin{pmatrix} \longleftarrow & v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_p & \longrightarrow \end{pmatrix}}_{p \times q \text{ matrix } V^T}$$

- $u_1, \ldots, u_p$ are orthonormal vectors in $\mathbb{R}^p$
- $v_1, \ldots, v_p$ are orthonormal vectors in $\mathbb{R}^q$
- $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ are **singular values**

Concisely approximating a matrix

Singular value decomposition of $p \times q$ matrix M (with $p \leq q$):

$$M = \begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_p \\ \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_p \end{pmatrix} \begin{pmatrix} \longleftarrow & v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_p & \longrightarrow \end{pmatrix}$$

A concise approximation to M, for any $k \leq p$

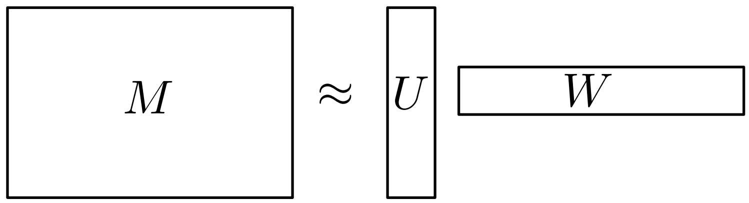
$$\widehat{M} = \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ u_1 & \cdots & u_k \\ \downarrow & & \downarrow \end{pmatrix}}_{p \times k} \underbrace{\begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_k \end{pmatrix}}_{k \times k} \underbrace{\begin{pmatrix} \longleftarrow & v_1 & \longrightarrow \\ & \vdots & \\ \longleftarrow & v_k & \longrightarrow \end{pmatrix}}_{k \times q}$$

Optimality property: Best low-rank approximation

Let M be any $p \times q$ matrix.

Want to approximate M by a $p \times q$ matrix $\widehat{M}$ of the form UW:

- U is $p \times k$ and W is $k \times q$
- $k \leq p, q$ is of our choosing



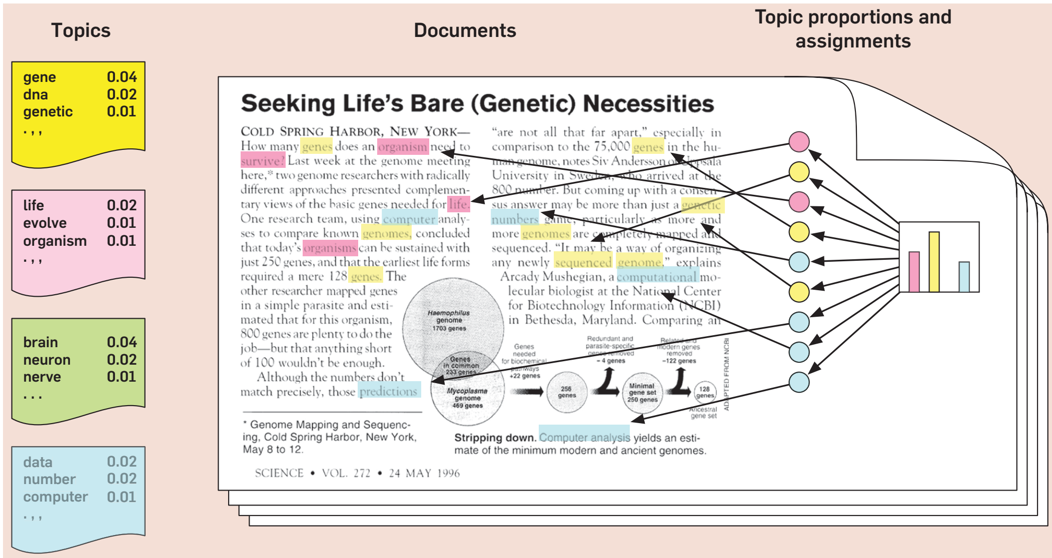
SVD yields the best such approximation $\widehat{M}$, minimizing the squared error

$$\sum_{i,j} (M_{ij} - \widehat{M}_{ij})^2.$$

B: Application to topic modeling

Topic modeling

Blei (2012):



Latent semantic indexing (LSI)

Given a large corpus of n documents:

- Fix a vocabulary, say of V words.
- Bag-of-words representation for documents: each document becomes a vector of length V, with one coordinate per word.
- The corpus is an $n \times V$ matrix, one row per document.

	cat	dog	house	boat	garden	...
Doc 1	4	1	1	0	2	
Doc 2	0	0	3	1	0	
Doc 3	0	1	3	0	0	
		⋮				

Let's find a concise approximation to this matrix M.

Latent semantic indexing, cont'd

Use SVD to get an approximation to M: for small k,

$$\underbrace{\begin{pmatrix} \longleftarrow \text{doc } 1 \longrightarrow \\ \longleftarrow \text{doc } 2 \longrightarrow \\ \longleftarrow \text{doc } 3 \longrightarrow \\ \vdots \\ \longleftarrow \text{doc } n \longrightarrow \end{pmatrix}}_{n \times V \text{ matrix } M} \approx \underbrace{\begin{pmatrix} \longleftarrow \theta_1 \longrightarrow \\ \longleftarrow \theta_2 \longrightarrow \\ \longleftarrow \theta_3 \longrightarrow \\ \vdots \\ \longleftarrow \theta_n \longrightarrow \end{pmatrix}}_{n \times k \text{ matrix } \Theta} \underbrace{\begin{pmatrix} \longleftarrow \quad \Psi_1 \quad \longrightarrow \\ \vdots \\ \longleftarrow \quad \Psi_k \quad \longrightarrow \end{pmatrix}}_{k \times V \text{ matrix } \Psi}$$

Latent semantic indexing, cont'd

$$\underbrace{\begin{pmatrix} \longleftarrow \text{doc } 1 \longrightarrow \\ \longleftarrow \text{doc } 2 \longrightarrow \\ \longleftarrow \text{doc } 3 \longrightarrow \\ \vdots \\ \longleftarrow \text{doc } n \longrightarrow \end{pmatrix}}_{n \times V \text{ matrix } M} \approx \underbrace{\begin{pmatrix} \longleftarrow \theta_1 \longrightarrow \\ \longleftarrow \theta_2 \longrightarrow \\ \longleftarrow \theta_3 \longrightarrow \\ \vdots \\ \longleftarrow \theta_n \longrightarrow \end{pmatrix}}_{n \times k \text{ matrix } \Theta} \underbrace{\begin{pmatrix} \longleftarrow \quad \Psi_1 \quad \longrightarrow \\ \vdots \\ \longleftarrow \quad \Psi_k \quad \longrightarrow \end{pmatrix}}_{k \times V \text{ matrix } \Psi}$$

Think of this as a *topic model* with k topics.

- Ψ_j is a vector of length V describing topic j: coefficient Ψ_{jw} is large if word w appears often in that topic.
- Each document is a combination of topics: θ_{ij} = weight of topic j in doc i.

Document i originally represented by ith row of M, a vector in $\mathbb{R}^V$.
Can instead use $\theta_i \in \mathbb{R}^k$, a more concise "semantic" representation.

Non-negative matrix factorization?

C: Application to recommender systems

Collaborative filtering

Details and images from Koren, Bell, Volinksy (2009).

Recommender systems: matching customers with products.

- Given: data on prior purchases/interests of users
- Recommend: further products of interest

Prototypical example: Netflix.

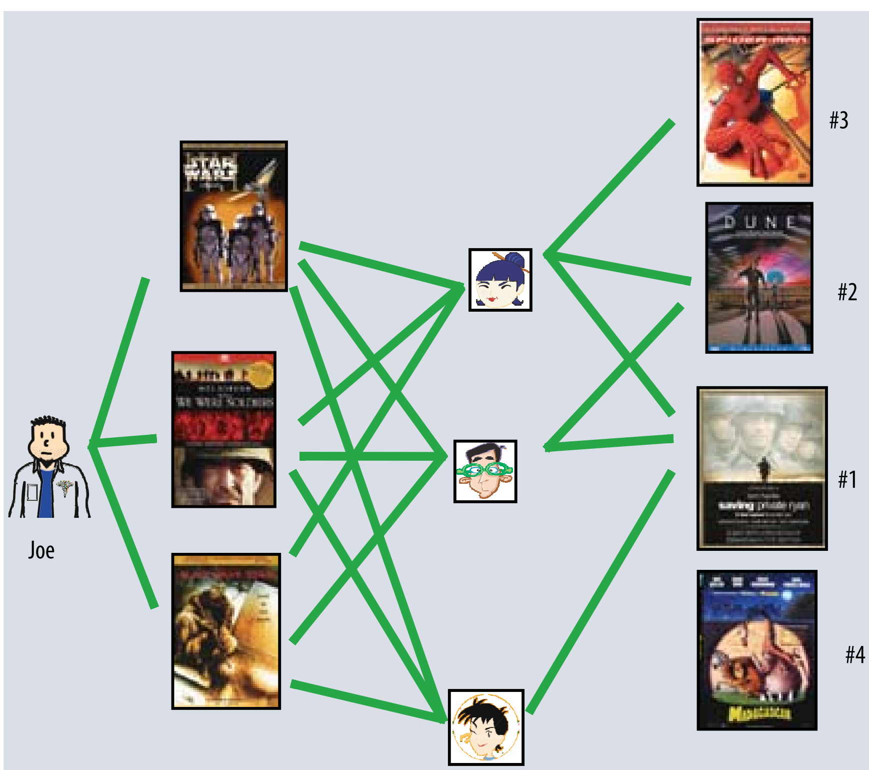
A successful approach: **collaborative filtering**.

- Model dependencies between different products, and between different users.
- Can give reasonable recommendations to a relatively new user.

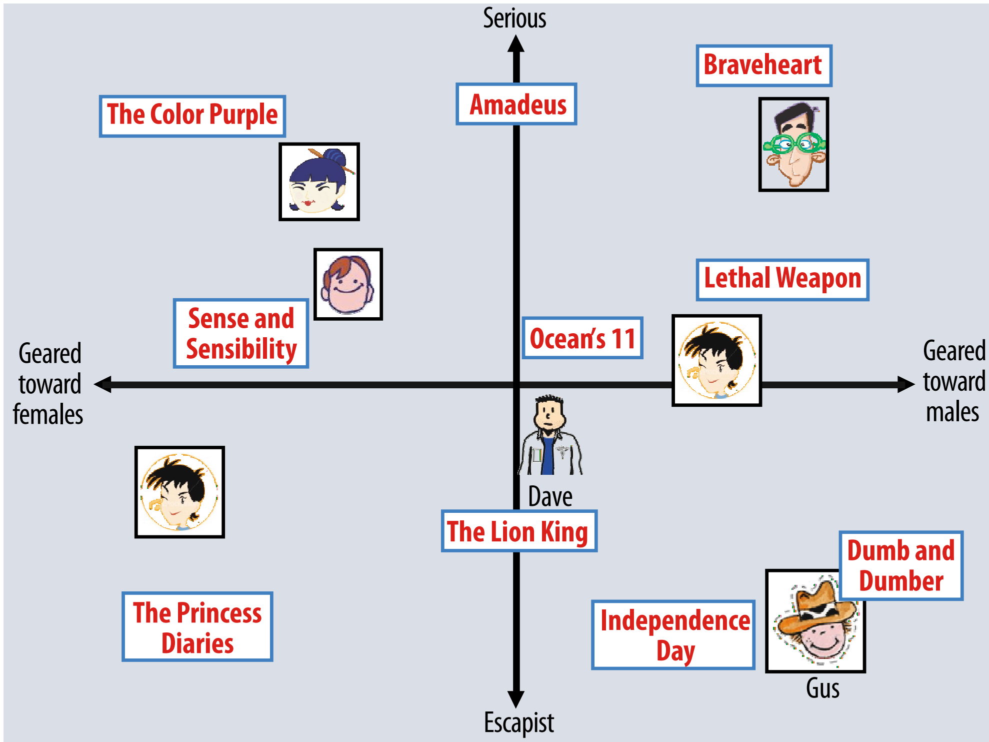
Two strategies for collaborative filtering:

- Neighborhood methods
- Latent factor methods

Neighborhood methods



Latent factor methods



The matrix factorization approach

User ratings are assembled in a large matrix M:

	Star Wars	Matrix	Casablanca	Camelot	Godfather	...
User 1	5	5	2	0	0	
User 2	0	0	3	4	5	
User 3	0	0	5	0	0	
		$\vdots$				

- Not rated = 0, otherwise scores 1-5.
- For n users and p movies, this has size $n \times p$.
- Most of the entries are unavailable, and we'd like to predict these.

Idea: Get best low-rank approximation of M, and use to fill in the missing entries.

User and movie factors

Best rank-k approximation is of the form $M \approx UW^T$:

$$\underbrace{\begin{pmatrix} \longleftarrow \text{user } 1 \longrightarrow \\ \longleftarrow \text{user } 2 \longrightarrow \\ \longleftarrow \text{user } 3 \longrightarrow \\ \vdots \\ \longleftarrow \text{user } n \longrightarrow \end{pmatrix}}_{n \times p \text{ matrix } M} \approx \underbrace{\begin{pmatrix} \longleftarrow u_1 \longrightarrow \\ \longleftarrow u_2 \longrightarrow \\ \longleftarrow u_3 \longrightarrow \\ \vdots \\ \longleftarrow u_n \longrightarrow \end{pmatrix}}_{n \times k \text{ matrix } U} \underbrace{\begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ w_1 & w_2 & \cdots & w_p \\ \downarrow & \downarrow & & \downarrow \end{pmatrix}}_{k \times p \text{ matrix } W^T}$$

Thus user i's rating of movie j is approximated as

$$M_{ij} \approx u_i \cdot w_j$$

User and movie factors

$$\underbrace{\begin{pmatrix} \longleftarrow \text{user } 1 \longrightarrow \\ \longleftarrow \text{user } 2 \longrightarrow \\ \longleftarrow \text{user } 3 \longrightarrow \\ \vdots \\ \longleftarrow \text{user } n \longrightarrow \end{pmatrix}}_{n \times p \text{ matrix } M} \approx \underbrace{\begin{pmatrix} \longleftarrow u_1 \longrightarrow \\ \longleftarrow u_2 \longrightarrow \\ \longleftarrow u_3 \longrightarrow \\ \vdots \\ \longleftarrow u_n \longrightarrow \end{pmatrix}}_{n \times k \text{ matrix } U} \underbrace{\begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ w_1 & w_2 & \cdots & w_p \\ \downarrow & \downarrow & & \downarrow \end{pmatrix}}_{k \times p \text{ matrix } W^T}$$

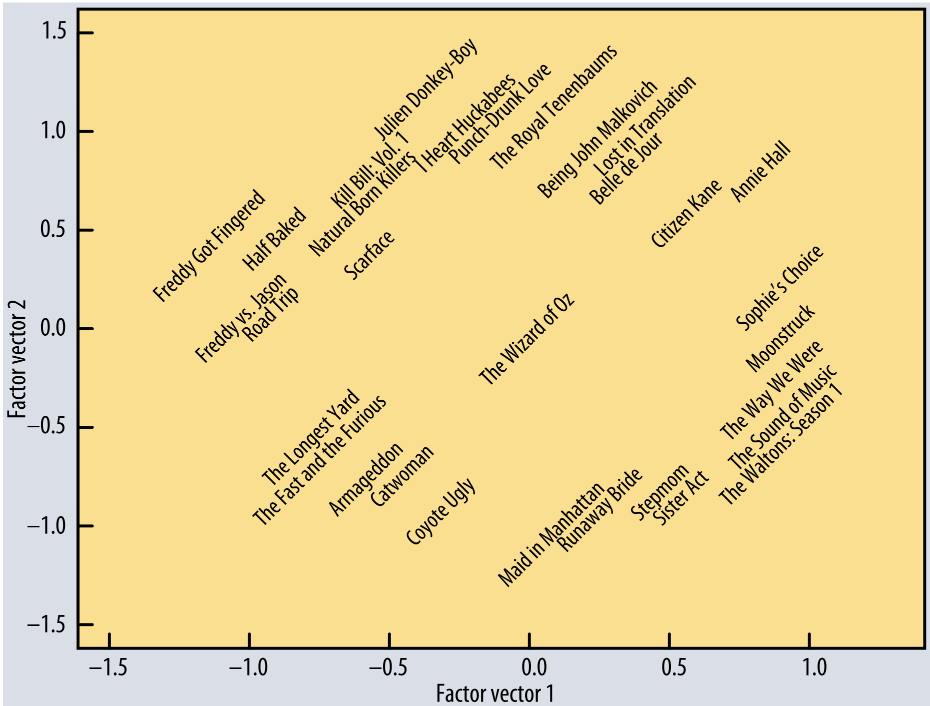
Thus user i's rating of movie j is approximated as

$$M_{ij} \approx u_i \cdot w_j$$

"Latent" representation embeds users and movies in the same k-dimensional space:

- Represent ith user by $u_i \in \mathbb{R}^k$
- Represent jth movie by $w_j \in \mathbb{R}^k$

Top two Netflix factors



Weighted singular value decomposition?