Randomized SVD

CSCI 4360/6360 Data Science II

Singular Value Decomposition (SVD)

- Given a (any!) matrix *M*, which is *n x m*, it can be represented as

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

- *U*: *n x n*, unitary matrix (orthogonal)

- Σ: *n x m*, diagonal matrix of singular values
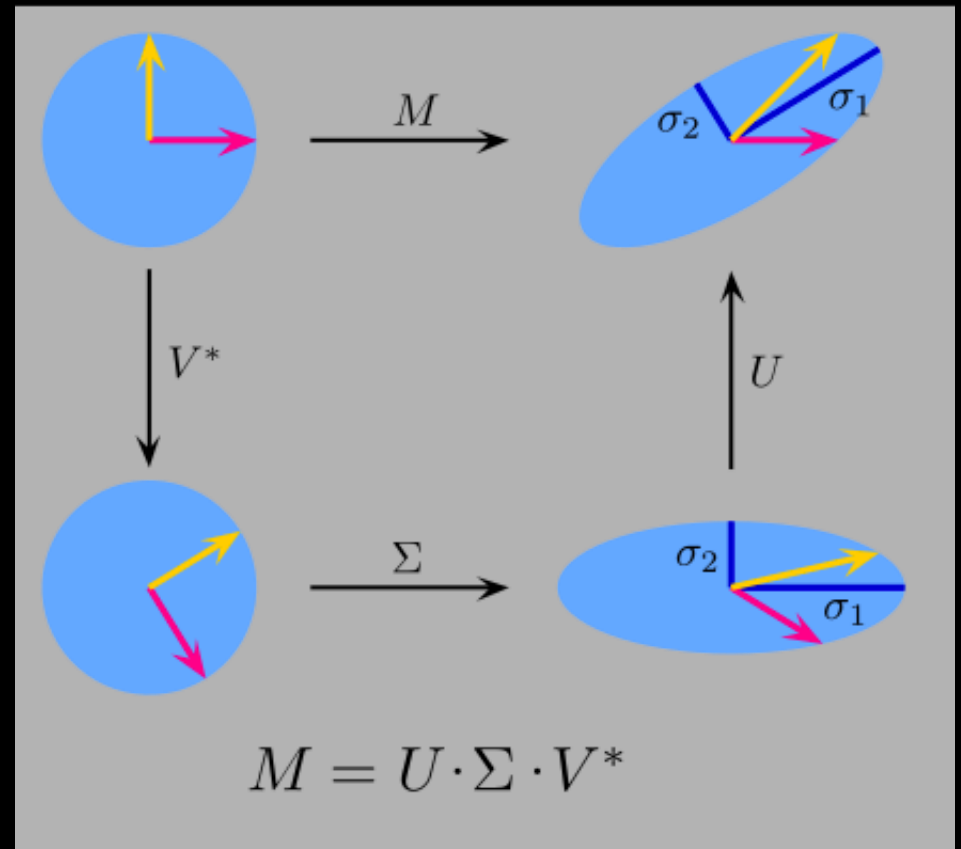
- *V^T*: *m x m*, unitary matrix (orthogonal)

SVD

$$A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T$$

- **A**: **Input data matrix**
  - $m \times n$ matrix (e.g., m documents, n terms)
- **U**: **Left singular vectors**
  - $m \times r$ matrix (m documents, r concepts)
- **Σ**: **Singular values**
  - $r \times r$ diagonal matrix (strength of each 'concept') (r : rank of the matrix **A**)
- **V**: **Right singular vectors**
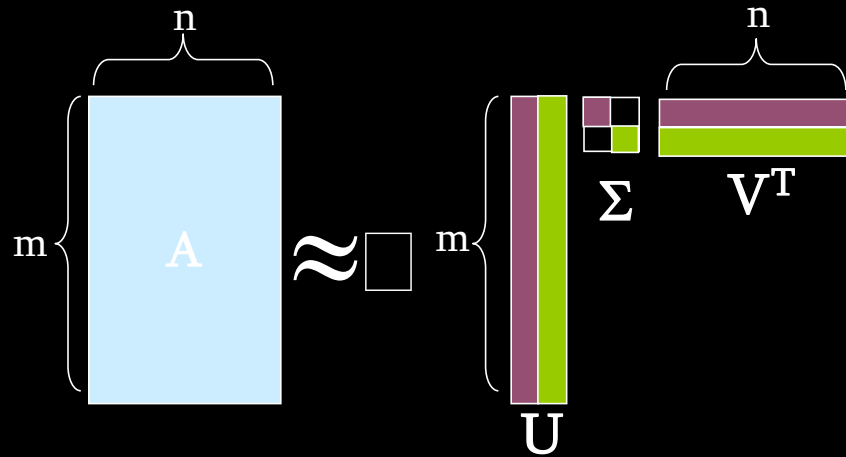 - $n \times r$ matrix (n terms, r concepts)

Singular Value Decomposition (SVD)

- Columns of *U* and *V* are orthonormal bases
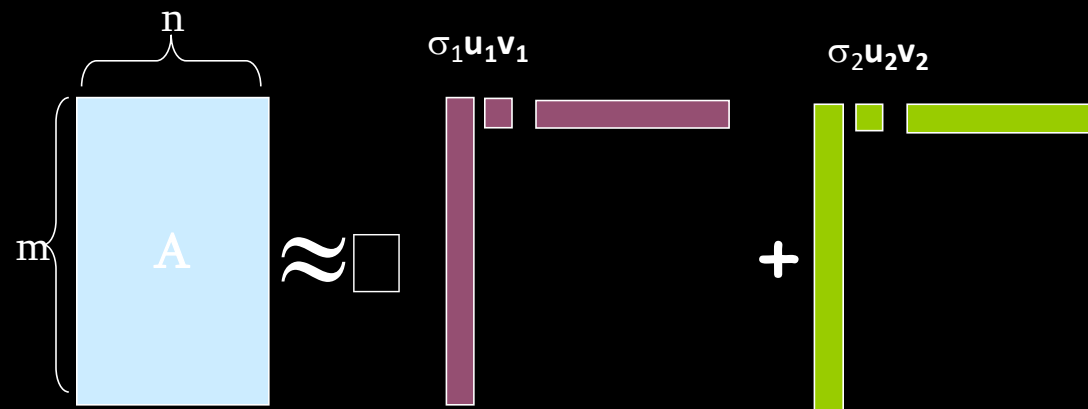- Singular values are the "strength" of each singular vector

SVD

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

SVD

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

σ_i … scalar
$\mathbf{u}_i$ … vector
$\mathbf{v}_i$ … vector

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

6

SVD - Properties

It is **always** possible to decompose a real matrix A into $A = U \Sigma V^T$, where

- U, Σ, V: unique

- U, V: column orthonormal
  - $U^T U = I$; $V^T V = I$ (I: identity matrix)
  - (Columns are orthogonal unit vectors)

- Σ: diagonal
  - Entries (**singular values**) are positive, and sorted in decreasing order ($\sigma_1 \geq \sigma_2 \geq ... \geq 0$)

Nice proof of uniqueness: http://www.mpi-inf.mpg.de/~bast/ir-seminar-ws04/lecture2.pdf
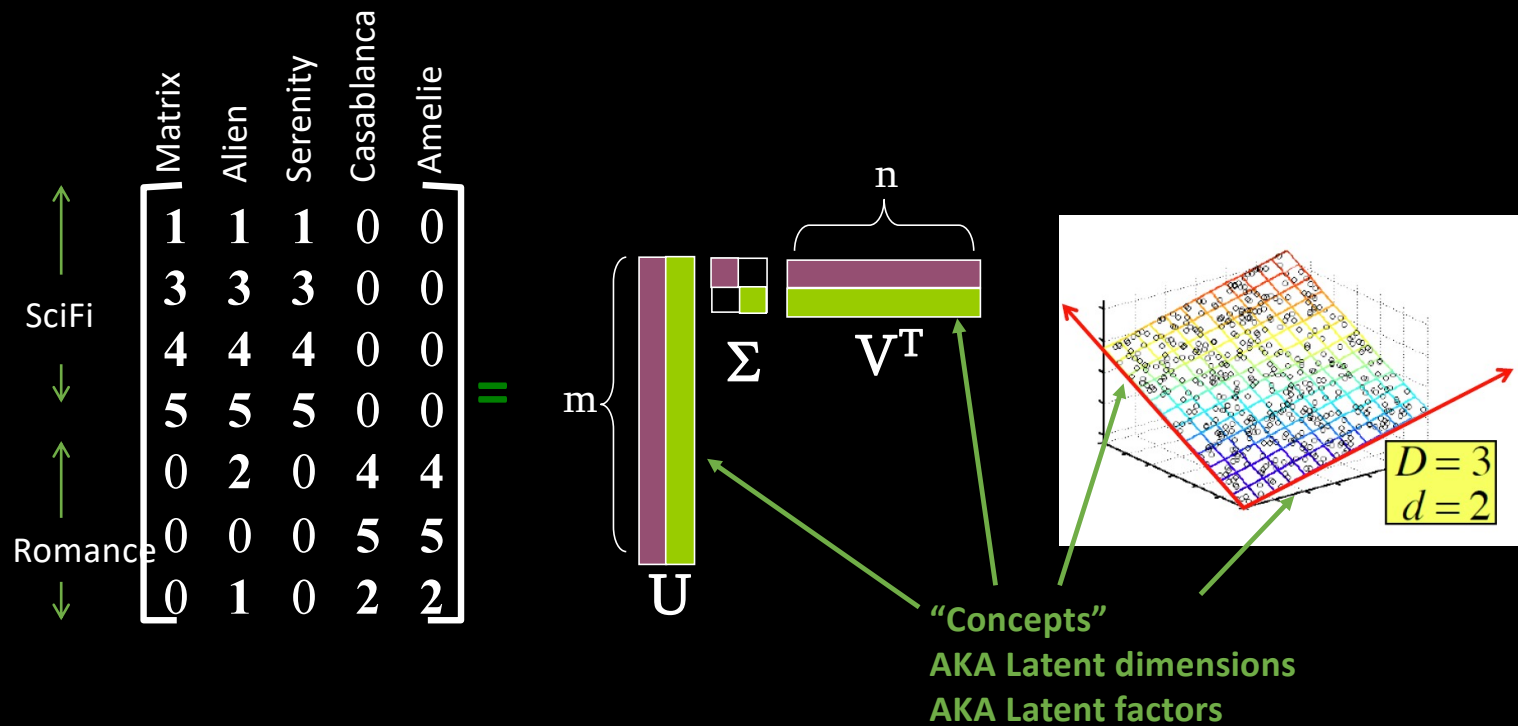
Why randomize SVD?

- **Runtime**
  - We're good at generating [pseudo-]random numbers
  - Can easily parallelize / distribute matrix algebra
  - SVD, like PCA, runs $O(n^3)$, making anything beyond ~10^3 infeasible

SVD – Example: Users-to-Movies

- **A = U Σ V^T - example: Users to Movies**

"Concepts"
AKA Latent dimensions
AKA Latent factors

SVD – Example: Users-to-Movies

- **A = U Σ V^T - example: Users to Movies**

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

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

Columns (matrix A): Matrix, Alien, Serenity, Casablanca, Amelie

Rows: SciFi, Romnce

SVD – Example: Users-to-Movies

- $A = U \Sigma V^T$ - example: Users to Movies

SciFi-concept

Romance-concept

	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi	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
Romnce	0	0	0	5	5
	0	1	0	2	2

=

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**

X

12.4	0	0
0	**9.5**	0
0	0	**1.3**

X

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

SVD – Example: Users-to-Movies

- **A = U Σ V^T - example:**   *U* is "user-to-concept" similarity matrix

SciFi-concept   Romance-concept

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

Matrix, Alien, Serenity, Casablanca, Amelie

SciFi, Romnce

12

SVD – Example: Users-to-Movies

- **A = U Σ V$^{\text{T}}$ - example:**

	Matrix	Alien	Serenity	Casablanca	Amelie
	1	1	1	0	0
	3	3	3	0	0
SciFi	4	4	4	0	0
	5	5	5	0	0
	0	2	0	4	4
Romnce	0	0	0	5	5
	0	1	0	2	2

=

SciFi-concept

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

X

"strength" of the SciFi-concept

12.4	0	0
0	9.5	0
0	0	1.3

X

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

SVD – Example: Users-to-Movies

- **A = U Σ Vᵀ - example:**

V is "movie-to-concept" similarity matrix

	Matrix	Alien	Serenity	Casablanca	Amelie
SciFi	1	1	1	0	0
	3	3	3	0	0
	4	4	4	0	0
	5	5	5	0	0
Romnce	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

=

SciFi-concept

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**

X

12.4	0	0
0	**9.5**	0
0	0	**1.3**

X

SciFi-concept

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

14

SVD - Interpretation #1

'**movies**', '**users**' and '**concepts**':

- U: user-to-concept similarity matrix

- V: movie-to-concept similarity matrix

- Σ: its diagonal elements:
    'strength' of each concept

SVD - Interpretation #2

More details

- **Q: How exactly is dim. reduction done?**

- **A: Set smallest singular values to zero**

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

SVD - Interpretation #2

More details

- **Q: How exactly is dim. reduction done?**

- **A: Set smallest singular values to zero**

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

SVD - Interpretation #2

More details

- **Q: How exactly is dim. reduction done?**
- **A: Set smallest singular values to zero**

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

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

18

SVD - Interpretation #2

More details

- **Q: How exactly is dim. reduction done?**

- **A: Set smallest singular values to zero**

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

SVD - Interpretation #2

More details

- **Q: How exactly is dim. reduction done?**

- **A: Set smallest singular values to zero**

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

$$\|A\text{-}B\|_F = \sqrt{\Sigma_{ij}\ (A_{ij}\text{-}B_{ij})^2}$$
is "small"

Frobenius norm:
$$\|M\|_F = \sqrt{\Sigma_{ij}\ M_{ij}{}^2}$$

SVD – Best Low Rank Approx.

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

B is best approximation of A

$$B = U \quad \text{Sigma} \quad V^T$$

Relationship to PCA

- SVD can be applied to *any* matrix; PCA only works on symmetric covariance matrices
- However, there is a relationship

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

- Columns of *V* are eigenvectors of $M^T M$
- Columns of *U* are eigenvectors of $M M^T$
- Singular values are square roots of eigenvalues of $M^T M$ or $M M^T$

SVD: Drawbacks

+**Optimal low-rank approximation**
in terms of Frobenius norm

−**Interpretability problem:**
- A singular vector specifies a linear combination of all input columns or rows

−**Lack of sparsity:**
- Singular vectors are **dense!**
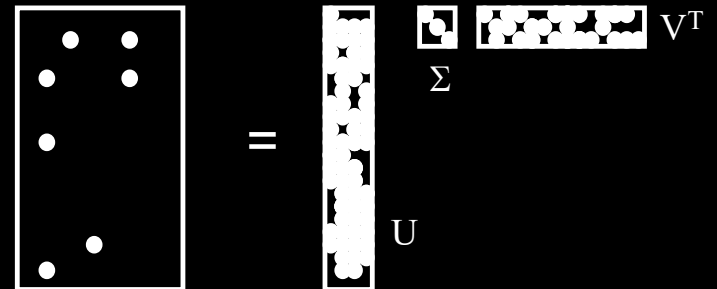
$$= U \; \Sigma \; V^T$$

CUR Decomposition

- **Goal: Express A as a product of matrices C,U,R**

  Make $\|A\text{-}C\cdot U\cdot R\|_F$ small

- **"Constraints" on C and R:**

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

| A | C | U | R |

CUR Decomposition

- **Goal: Express A as a product of matrices C,U,R**

  Make $\|A\text{-}C\cdot U\cdot R\|_F$ small

- **"Constraints" on C and R:**

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

**Pseudo-inverse of
the intersection of C and R**
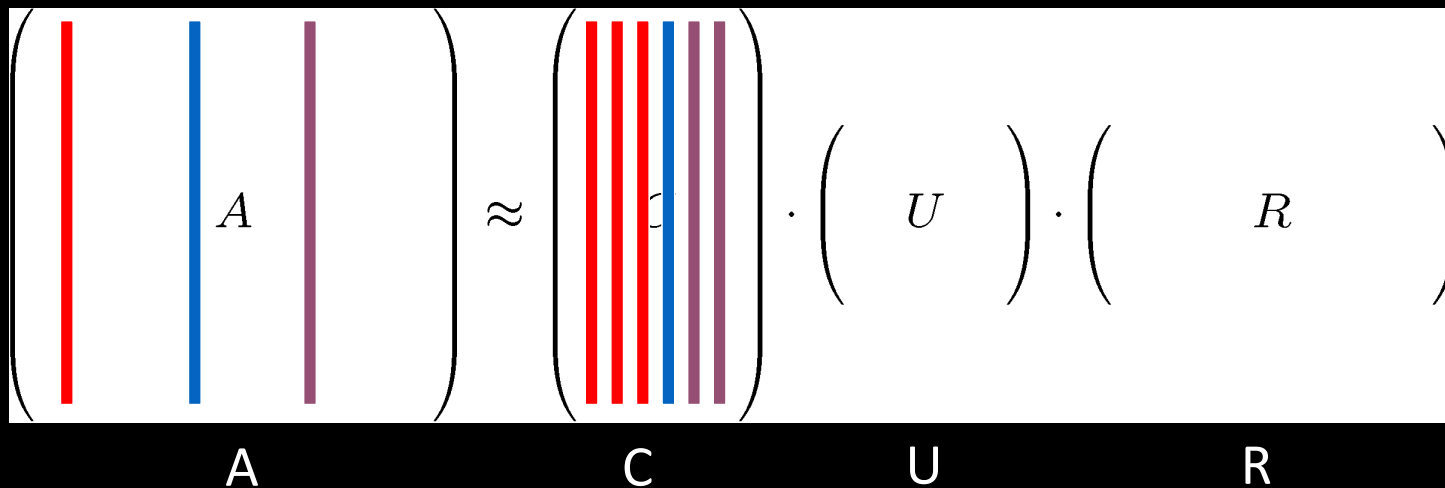
A   C   U   R

CUR: How it Works

Note this is a randomized algorithm; the same column can be sampled more than once
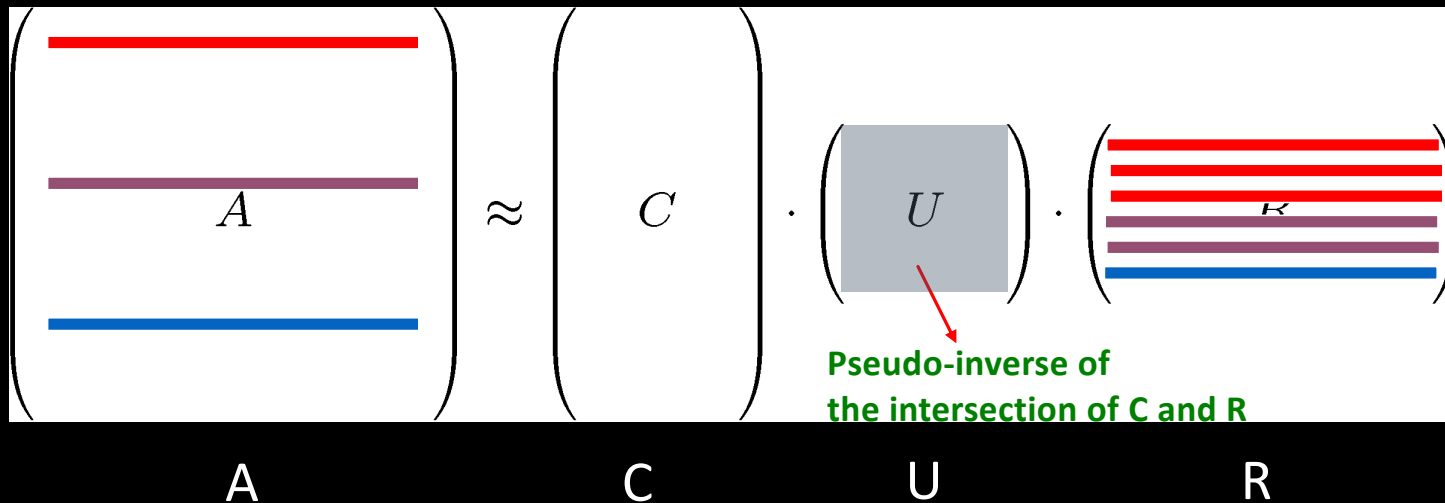
- **Sampling columns (similarly for rows):**

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 $\mathbf{C}_d(:, i) = \mathbf{A}(:, j) / \sqrt{cP(j)}$

Computing U

- Let **W** be the "intersection" of sampled columns **C** and rows **R**
  - Let SVD of **W** = **X Z Y**^{T}

- **Then: U = W$^+$ = Y Z$^+$ X^T**
  - **Z$^+$: reciprocals of non-zero singular values:** $Z^+_{ii} = 1/Z_{ii}$
  - W$^+$ is the "**pseudoinverse**"

$$U = W^+$$
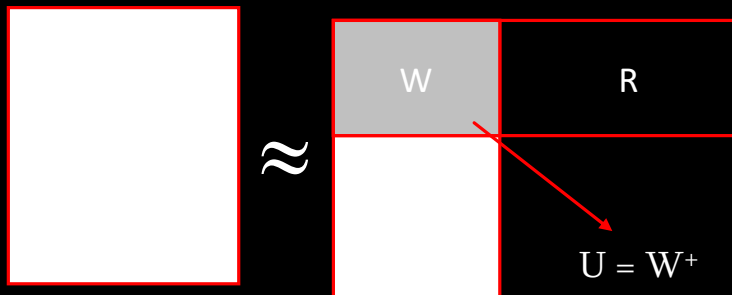
Why pseudoinverse works?
W = X Z Y then W^{-1} = X^{-1} Z^{-1} Y^{-1}
Due to orthonomality
X^{-1}=X^T and Y^{-1}=Y^T
Since Z is diagonal Z^{-1} = 1/Z$_{ii}$
Thus, if **W** is nonsingular, pseudoinverse is the true inverse

CUR: Pros & Cons

+Easy interpretation

- Since the basis vectors are actual columns and rows

+Sparse basis

- Since the basis vectors are actual columns and rows

-Duplicate columns and rows

- Columns of large norms will be sampled many times

Actual column
Singular vector
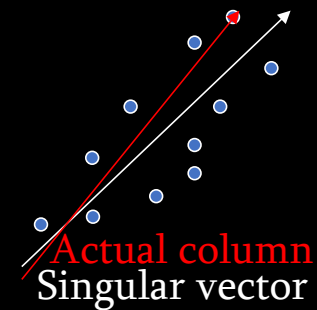
Solution

- **If we want to get rid of the duplicates:**
  - Throw them away
  - Scale (multiply) the columns/rows by the square root of the number of duplicates

Construct a small U
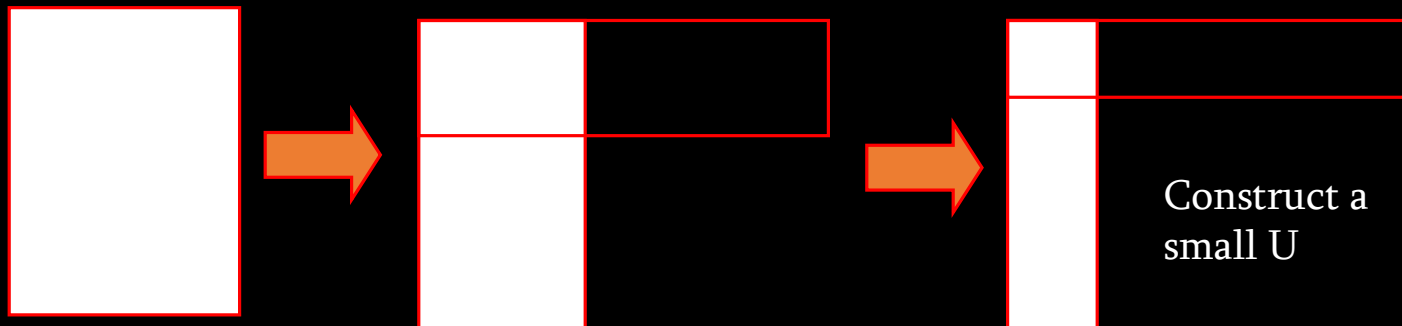
SVD vs. CUR

sparse and small

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

Huge but sparse   Big and dense

dense but small

CUR: $A = C U R$

Huge but sparse   Big but sparse

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Stochastic SVD (SSVD)

- Uses **random projections** to find close approximation to SVD
- Combination of probabilistic strategies to maximize convergence likelihood
- Easily scalable to *massive* linear systems

Basic goal

- Matrix *A*
  - Find a low-rank approximation of *A*
  - Basic dimensionality reduction

$$\|A - QQ^*A\| < \epsilon$$

Preconditioning

Approximating range of *A*

- INPUT: *A, k, p*
- OUTPUT: *Q*

1. Draw Gaussian n x k test matrix Ω
2. Form product Y = AΩ
3. Orthogonalize columns of Y ➜ Q

Approximating SVD of *A*

- INPUT: *Q*
- OUTPUT: Singular vectors *U*

1. Form k x n matrix B = Q^{T}A
2. Compute SVD of B = ÛΣV^T
3. Compute singular vectors U = QÛ

Empirical Results

- 1000x1000 matrix

- Several runs of empirical results (blue) to theoretical lower bound (red)

- **Error seems to be systemic**
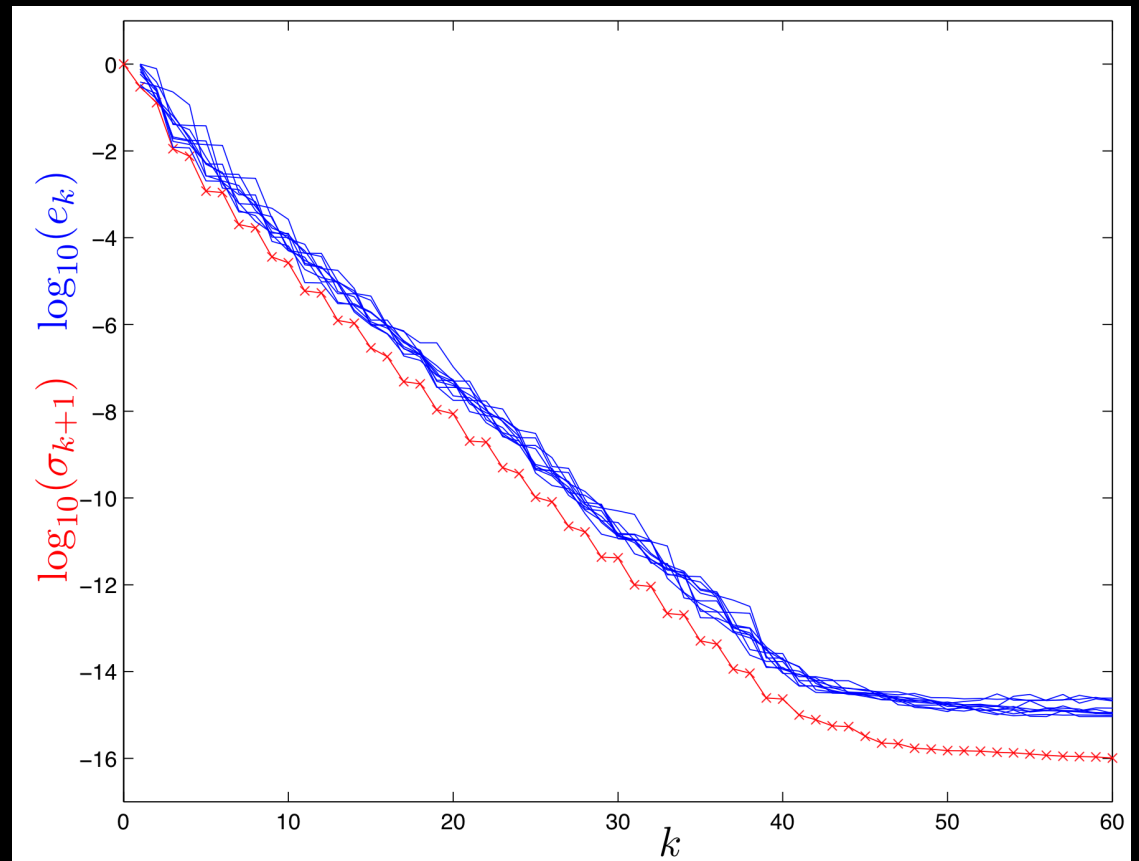
Power iterations

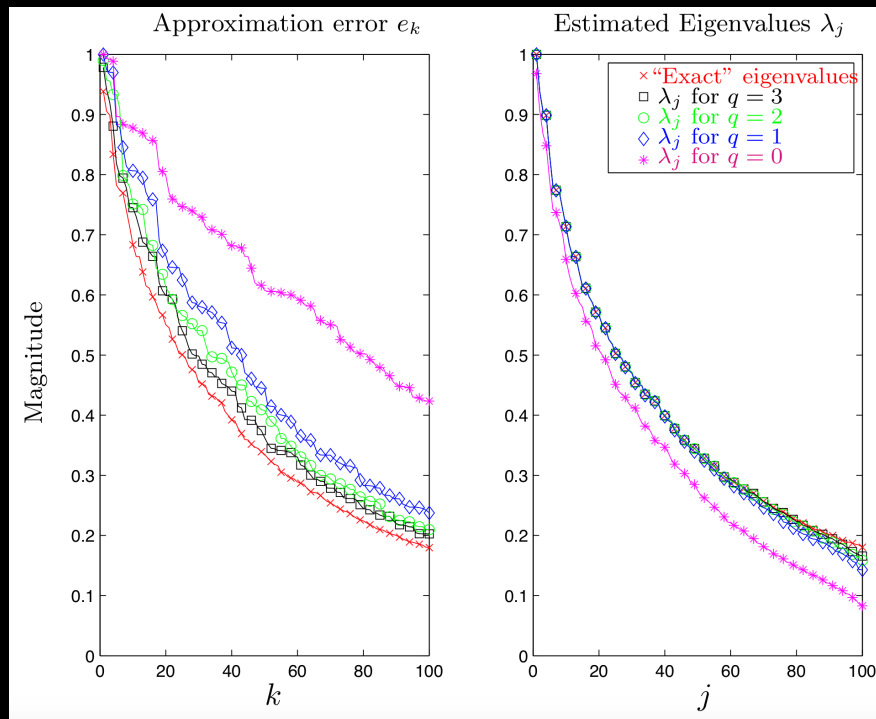- Affects decay of eigenvalues / singular values

$$Y = \times \Omega.$$

$$Y = (A A^*)^q A \Omega$$

Power iterations

$$\mathbb{E}\|A - QQ^{\mathsf{T}}A\|_2 \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1} + \frac{e\sqrt{k+p}}{p} \cdot \left(\sum_{j>k}\sigma_j^2\right)^{1/2}$$

$$\leq \left[1 + \frac{4\sqrt{k+p}}{p-1} \cdot \sqrt{\min\{m, n\}}\right]\sigma_{k+1}$$

$$= C \cdot \sigma_{k+1}.$$

Upshot: after only a single power iteration, the error is proportional to the *next* [uncomputed] singular value (times a constant *C*).

Empirical Results
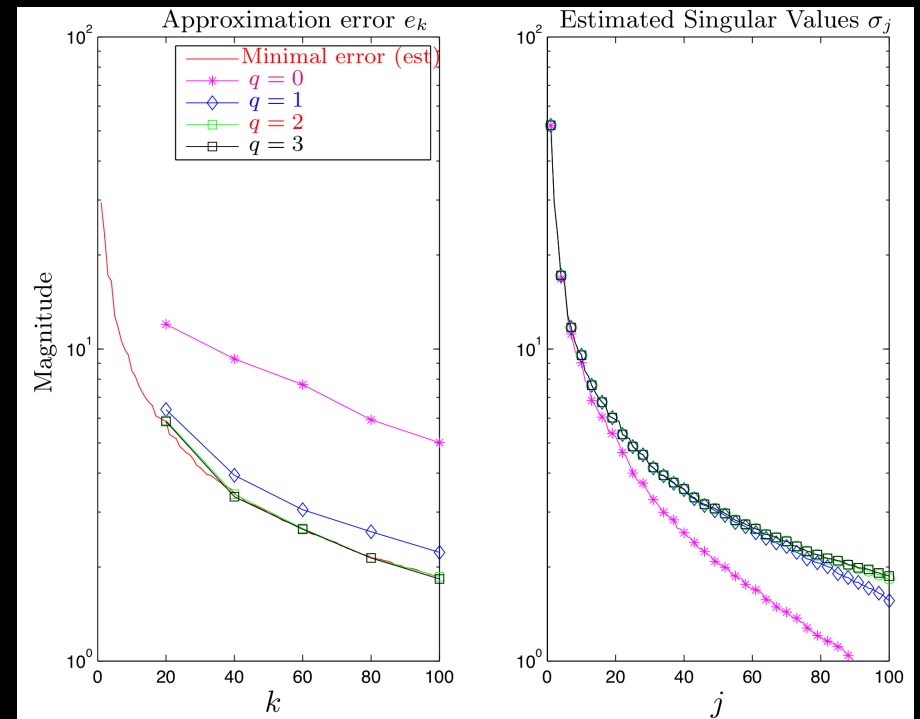
Why does this work?

- Three primary reasons:

1. **Johnson-Lindenstrauss Lemma**
   - Low-dimensional embeddings preserve pairwise distances

$$(1 - \varepsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \varepsilon)\|u - v\|^2$$

2. **Concentration of measure**
   - Geometric interpretation of classical idea: regular functions of independent random variables rarely deviate far from their means

3. **Preconditioning**
   - Condition number: how much change in output is produced from change in input (relation to #1)
   - *Q* matrix lowers condition number while preserving overall system

$$\kappa = \frac{|\lambda_{\max}|}{|\lambda_{\min}|}$$

Summary

- Relationship of SVD and PCA
  - PCA: eigenvectors and eigenvalues of the covariance matrix (or kernel matrix, for Kernel PCA)
  - SVD: Low-rank approximation for *any* matrix
- CUR
  - Randomly sample columns of data matrix *A* to use as basis
  - Interpretable and sparse, but potentially oversample high-magnitude columns
- SVD via SGD
  - Reframe SVD as a matrix completion problem
  - Use SGD in alternating least-squares to infer "missing" components
- SSVD
  - Full-blown Johnson-Lindenstrauss exploitation
  - Use random projections to approximate SVD to high accuracy
  - Requires some empirical tweaks (oversampling, power iterations)

References

- "Randomized methods for computing low-rank approximations of matrices", https://amath.colorado.edu/faculty/martinss/Pubs/2012_halko_dissertation.pdf

- "CUR decomposition for compression and compressed sensing of large-scale traffic data", https://dspace.mit.edu/openaccess-disseminate/1721.1/86879

Collaborative Filtering

- ▶ Problem
  - ▶ Set of users
  - ▶ Set of items (movies, books, jokes, products, stories, ...)
  - ▶ Feedback (ratings, purchase, click-through, tags, ...)
- ▶ Predict additional items a user may like
  - ▶ Assumption: Similar feedback $\implies$ Similar taste
- ▶ Example

$$
\begin{array}{c c c c}
 & Avatar & The\ Matrix & Up \\
Alice & ? & 4 & 2 \\
Bob & 3 & 2 & ? \\
Charlie & 5 & ? & 3
\end{array}
$$
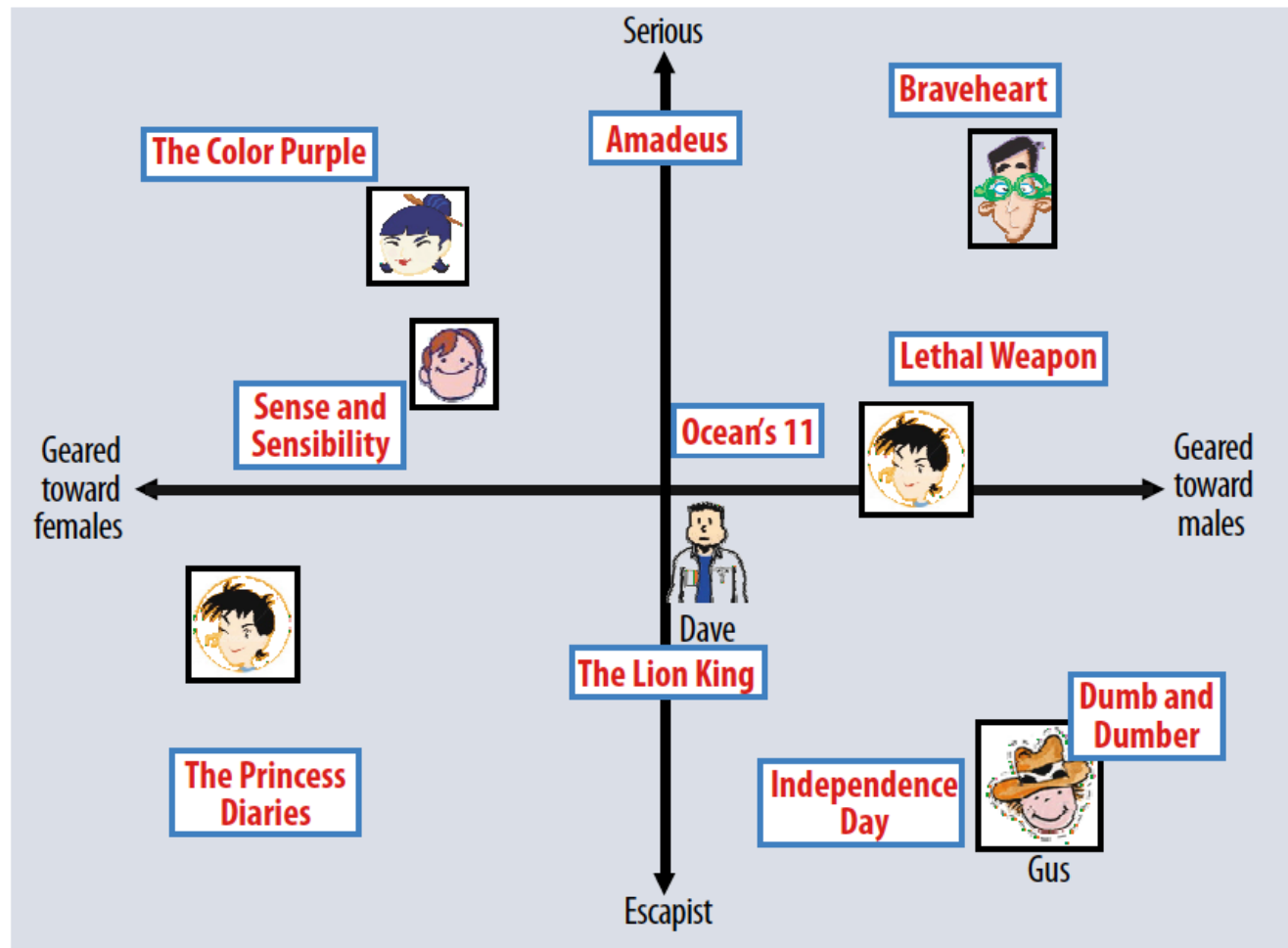
- ▶ Netflix competition: 500k users, 20k movies, 100M movie ratings, 3M question marks

Recovering latent factors in a matrix
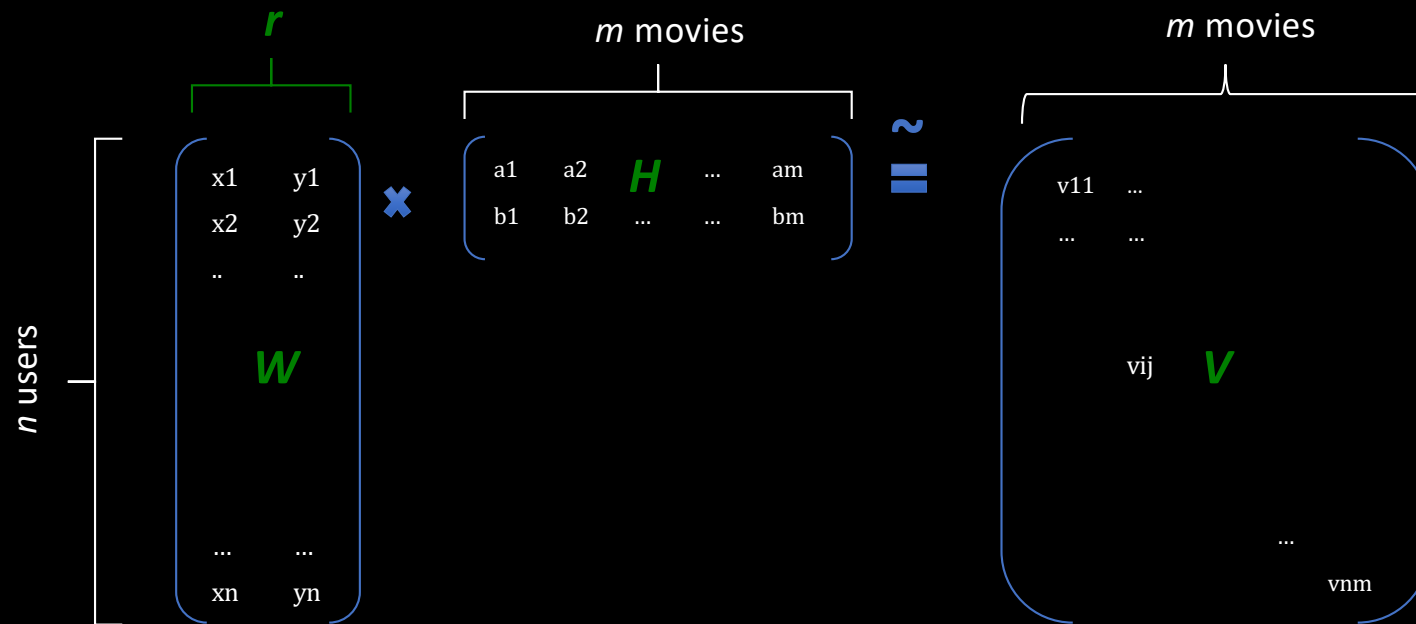
V[i,j] = user i's rating of movie j

Semantic Factors (Koren et al., 2009)

Latent Factor Models

▶ Discover latent factors ($r = 1$)

	Avatar (2.24)	The Matrix (1.92)	Up (1.18)
Alice (1.98)		4 (3.8)	2 (2.3)
Bob (1.21)	3 (2.7)	2 (2.3)	
Charlie (2.30)	5 (5.2)		3 (2.7)

▶ Minimum loss

$$\min_{\mathbf{W},\mathbf{H}} \sum_{(i,j)\in Z} (\mathbf{V}_{ij} - [\mathbf{WH}]_{ij})^2$$

Latent Factor Models

- Discover latent factors $(r = 1)$

	Avatar (2.24)	The Matrix (1.92)	Up (1.18)
Alice (1.98)	? (4.4)	4 (3.8)	2 (2.3)
Bob (1.21)	3 (2.7)	2 (2.3)	? (1.4)
Charlie (2.30)	5 (5.2)	? (4.4)	3 (2.7)

- Minimum loss

$$\min_{\mathbf{W},\mathbf{H},\mathbf{u},\mathbf{m}} \sum_{(i,j)\in Z} (\mathbf{V}_{ij} - \mu - \mathbf{u}_i - \mathbf{m}_j - [\mathbf{WH}]_{ij})^2$$
$$+ \lambda \left(\|\mathbf{W}\| + \|\mathbf{H}\| + \|\mathbf{u}\| + \|\mathbf{m}\| \right)$$

- Bias, regularization

Matrix factorization as SGD

require that the loss can be written as

$$L = \sum_{(i,j) \in Z} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$$

Algorithm 1 SGD for Matrix Factorization

Require: A training set Z, initial values $\boldsymbol{W}_0$ and $\boldsymbol{H}_0$
　　while not converged **do** {step}
　　　　Select a training point $(i, j) \in Z$ uniformly at random.
　　　　$\boldsymbol{W}'_{i*} \leftarrow \boldsymbol{W}_{i*} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{W}_{i*}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
　　　　$\boldsymbol{H}_{*j} \leftarrow \boldsymbol{H}_{*j} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{H}_{*j}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
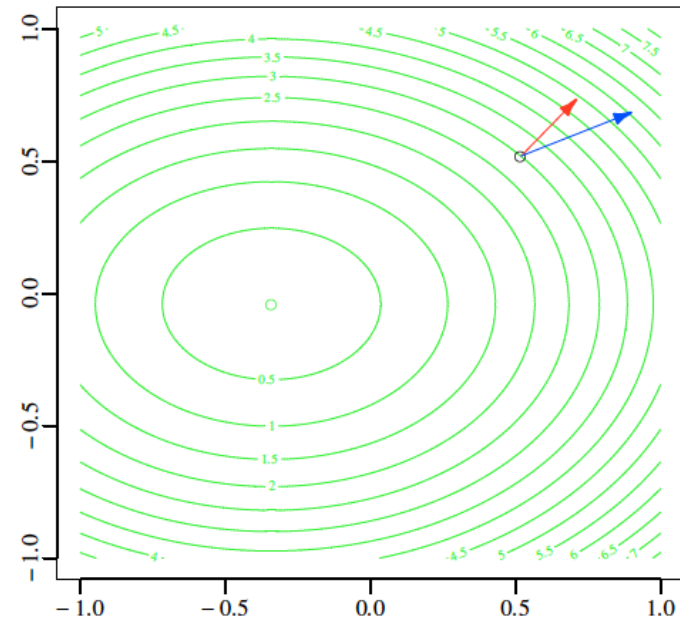　　　　$\boldsymbol{W}_{i*} \leftarrow \boldsymbol{W}'_{i*}$
　　end while

---

why does this work?

Stochastic Gradient Descent

▶ Find minimum θ^* of function L

▶ Pick a starting point θ_0
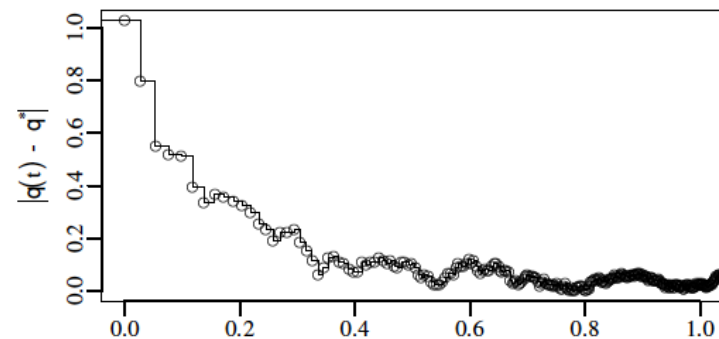
▶ Approximate gradient $\hat{L}'(\theta_0)$

Stochastic Gradient Descent

- Find minimum θ^* of function L
- Pick a starting point θ_0
- Approximate gradient $\hat{L}'(\theta_0)$
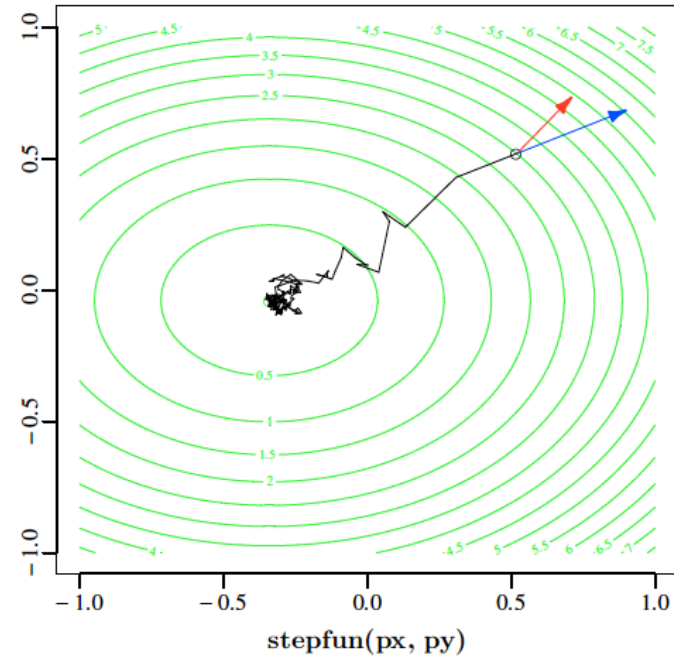- Jump "approximately" downhill
- Stochastic difference equation

$$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n)$$

- Under certain conditions, asymptotically approximates (continuous) gradient descent

Why does this work?

require that the loss can be written as

$$L = \sum_{(i,j) \in Z} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$$

Algorithm 1 SGD for Matrix Factorization

Require: A training set Z, initial values $\boldsymbol{W}_0$ and $\boldsymbol{H}_0$
    while not converged **do** {step}
        Select a training point $(i, j) \in Z$ uniformly at random.
        $\boldsymbol{W}'_{i*} \leftarrow \boldsymbol{W}_{i*} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{W}_{i*}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
        $\boldsymbol{H}_{*j} \leftarrow \boldsymbol{H}_{*j} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{H}_{*j}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$
        $\boldsymbol{W}_{i*} \leftarrow \boldsymbol{W}'_{i*}$
    end while

Key Claim

require that the loss can be written as
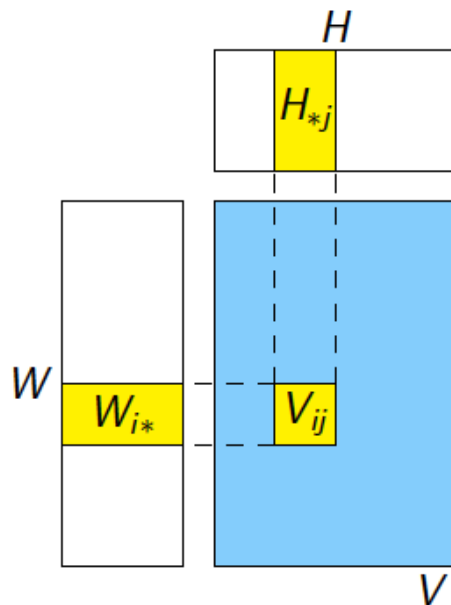
$$L = \sum_{(i,j) \in Z} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$$

$$\frac{\partial}{\partial \boldsymbol{W}_{i'k}} L_{ij}(\boldsymbol{W}, \boldsymbol{H}) = \begin{cases} 0 & \text{if } i \neq i' \\ \frac{\partial}{\partial \boldsymbol{W}_{ik}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j}) & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial \boldsymbol{H}_{kj'}} L_{ij}(\boldsymbol{W}, \boldsymbol{H}) = \begin{cases} 0 & \text{if } j \neq j' \\ \frac{\partial}{\partial \boldsymbol{H}_{kj}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j}) & \text{otherwise} \end{cases}$$

Checking the claim

$$\frac{\partial}{\partial W_{i*}} L(W, H) = \frac{\partial}{\partial W_{i*}} \sum_{(i',j) \in Z} L_{i'j}(W_{i'*}, H_{*j}) = \sum_{j \in Z_{i*}} \frac{\partial}{\partial W_{i*}} L_{ij}(W_{i*}, H_{*j}),$$

where $Z_{i*} = \{\, j : (i,j) \in Z \,\}.$

$$\frac{\partial}{\partial H_{*j}} L(W, H) = \sum_{i \in Z_{*j}} \frac{\partial}{\partial W_{*j}} L_{ij}(W_{i*}, H_{*j}),$$

where $Z_{*j} = \{\, i : (i,j) \in Z \,\}.$

Think for SGD for logistic regression
- LR loss = compare y and $\hat{y}$ = dot(w,x)
- similar but now update w (user weights) and x (movie weight)

Stochastic Gradient Descent on Netflix Data