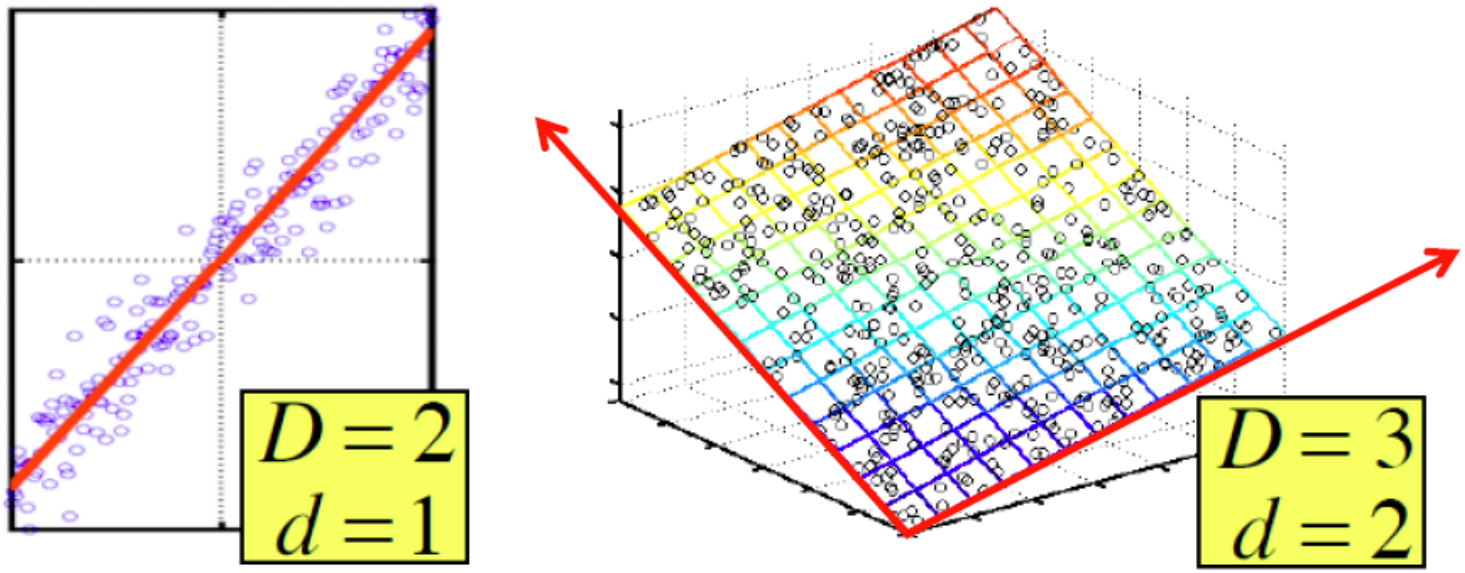
Dimensionality Reduction

Shannon Quinn

(with thanks to William Cohen of Carnegie Mellon University, and J. Leskovec, A. Rajaraman, and J. Ullman of Stanford University)

Dimensionality Reduction



- **Assumption:** Data lies on or near a low *d*-dimensional subspace
- **Axes of this subspace are effective representation of the data**

Rank of a Matrix

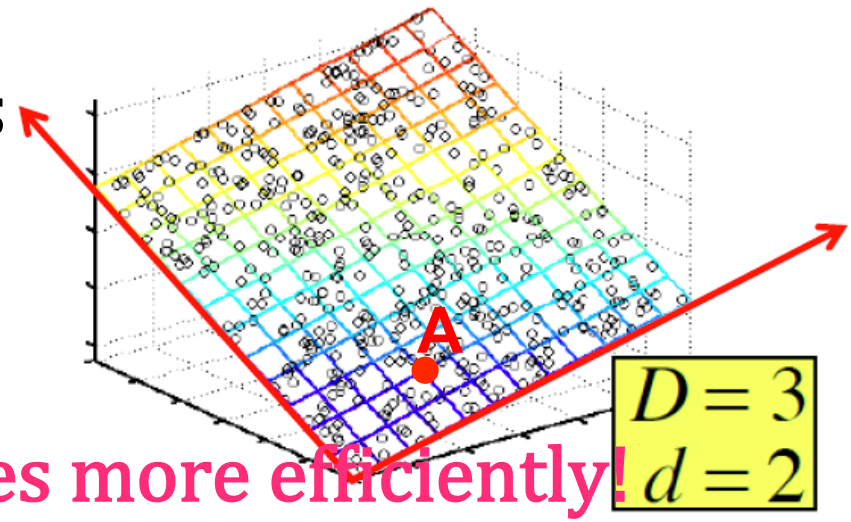
- Q: What is **rank** of a matrix **A**?
- A: Number of linearly independent columns of A
- **For example:**
  - Matrix **A** = $\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$ has rank **r=2**
    - **Why?** The first two rows are linearly independent, so the rank is at least 2, but all three rows are linearly dependent (the first is equal to the sum of the second and third) so the rank must be less than 3.
- **Why do we care about low rank?**
  - We can write **A** as two "basis" vectors: [1 2 1] [-2 -3 1]
  - And new coordinates of : [1 0] [0 1] [1 1]

Rank is "Dimensionality"

- Cloud of points 3D space:
  - Think of point positions as a matrix:

1 row per point:

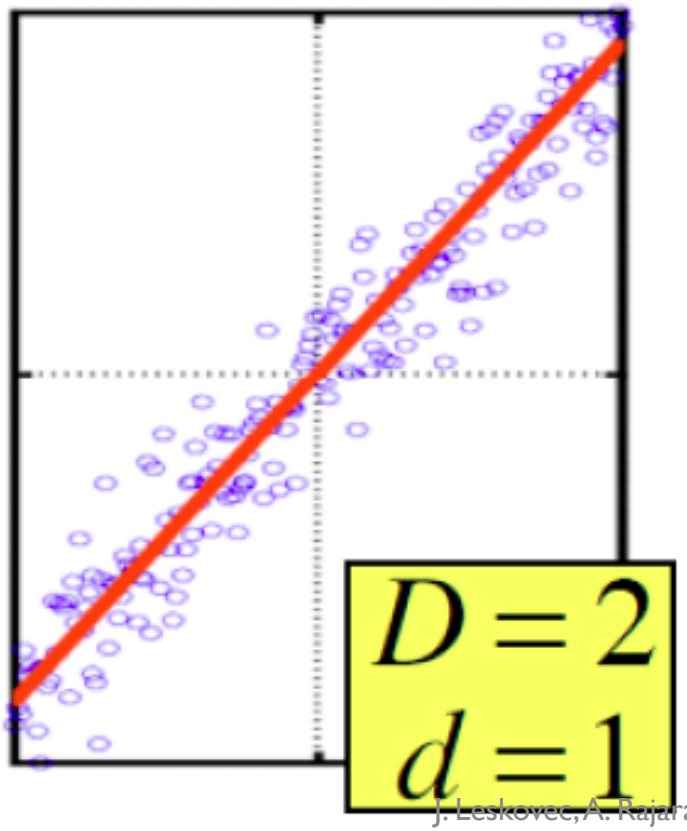
$$\begin{array}{c} \\ \\ \end{array}\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}\begin{array}{l} \text{A} \\ \text{B} \\ \text{C} \end{array}$$



- We can rewrite coordinates more efficiently!
  - Old basis vectors: [1 0 0] [0 1 0] [0 0 1]
  - **New basis vectors: [1 2 1] [-2 -3 1]**
  - Then **A** has new coordinates: [1 0]. **B**: [0 1], **C**: [1 1]
    - **Notice: We reduced the number of coordinates!**

Dimensionality Reduction

- Goal of dimensionality reduction is to discover the axis of data!



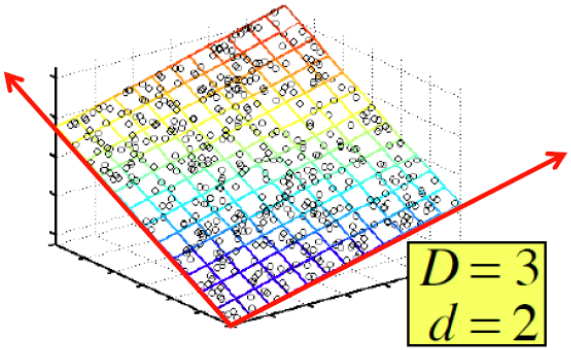
Rather than representing every point with 2 coordinates we represent each point with 1 coordinate (corresponding to the position of the point on the red line).

By doing this we incur a bit of **error** as the points do not exactly lie on the line

Why Reduce Dimensions?

Why reduce dimensions?

- Discover hidden correlations/topics
  - Words that occur commonly together
- Remove redundant and noisy features
  - Not all words are useful
- Interpretation and visualization
- Easier storage and processing of



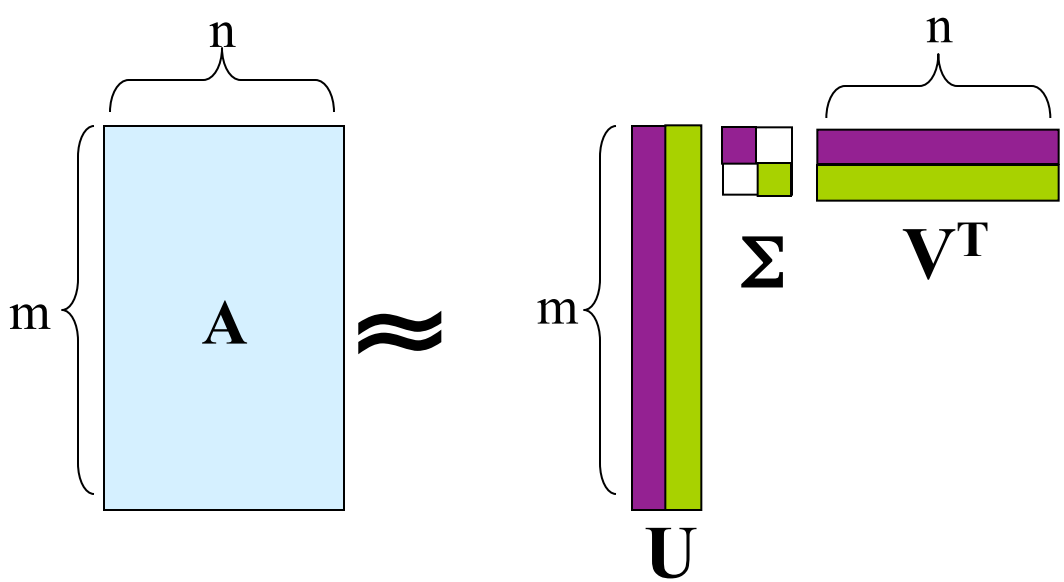
SVD - Definition

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

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

SVD

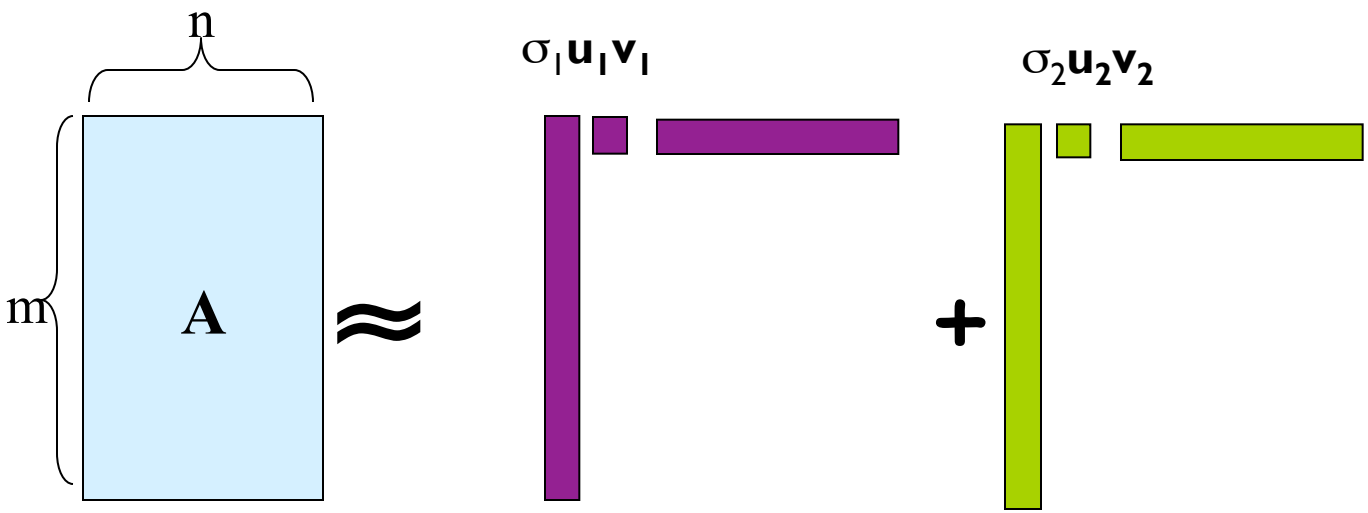
$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^T$$



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

SVD

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



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

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 ($\sigma_1 \geq \sigma_2 \geq \ldots \geq 0$)

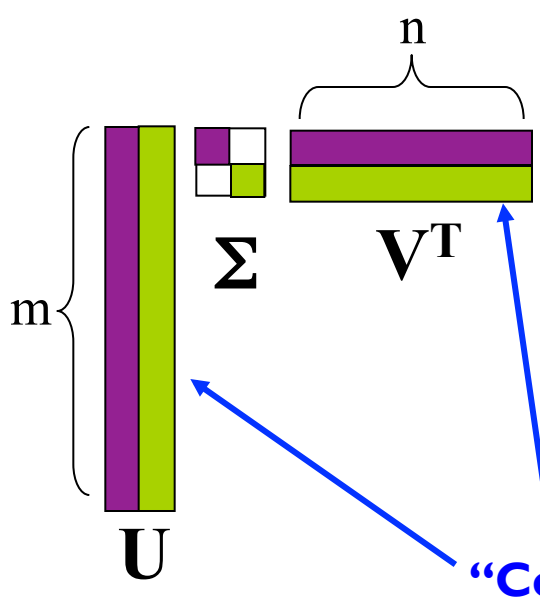
Nice proof of uniqueness: http://www.mpi-inf.mpg.de/~bast/ir-seminar-ws04/lecture2.pdf

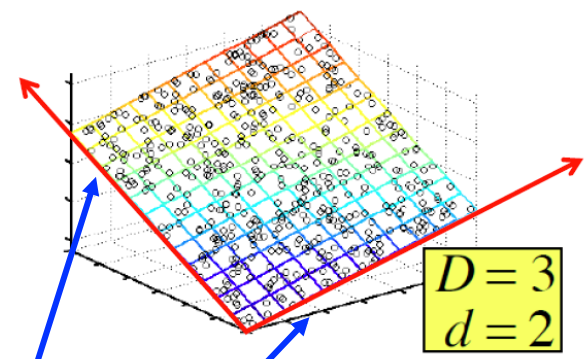
SVD – Example: Users-to-Movies

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

	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
Romance ↑	0	2	0	4	4
	0	0	0	5	5
↓	0	1	0	2	2

$=$





"Concepts"
AKA Latent dimensions
AKA Latent factors

SVD – Example: Users-to-Movies

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

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

SVD – Example: Users-to-Movies

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

SciFi-concept

Romance-concept

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

SVD – Example: Users-to-Movies

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

U **is "user-to-concept' similarity matrix**

SciFi-concept Romance-concept

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

SVD – Example: Users-to-Movies

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

SciFi-concept

"strength" of the SciFi-concept

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

SVD – Example: Users-to-Movies

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

V **is "movie-to-concept" similarity matrix**

SciFi-concept

SciFi-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
Romnce	0	2	0	4	4
	0	0	0	5	5
	0	1	0	2	2

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

SVD - 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

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

SVD - Interpretation #2

More details

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

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

SVD - Interpretation #2

More details

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

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

SVD - 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

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

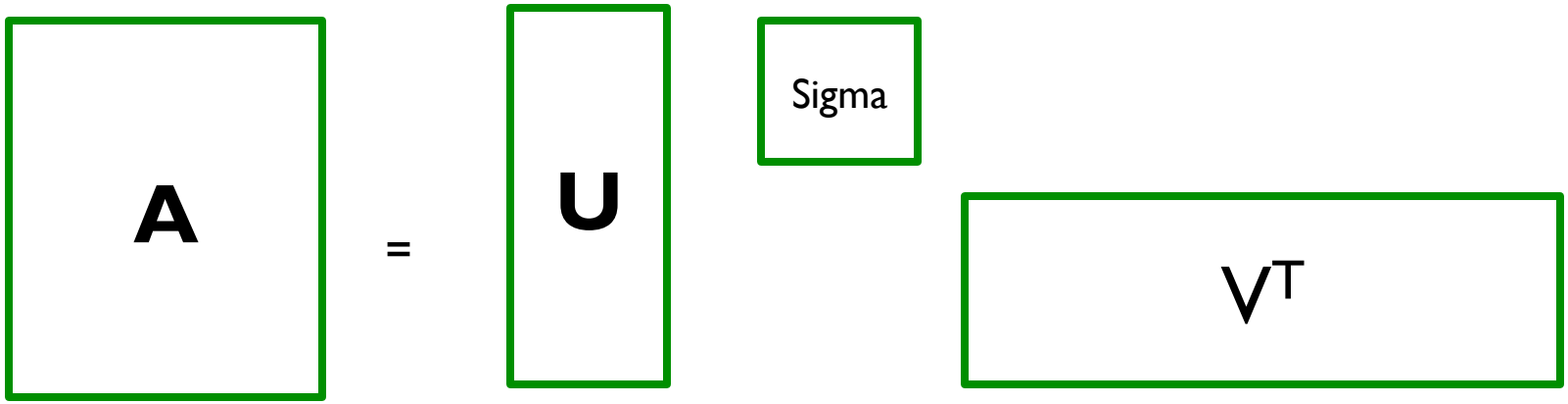
Frobenius norm:

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

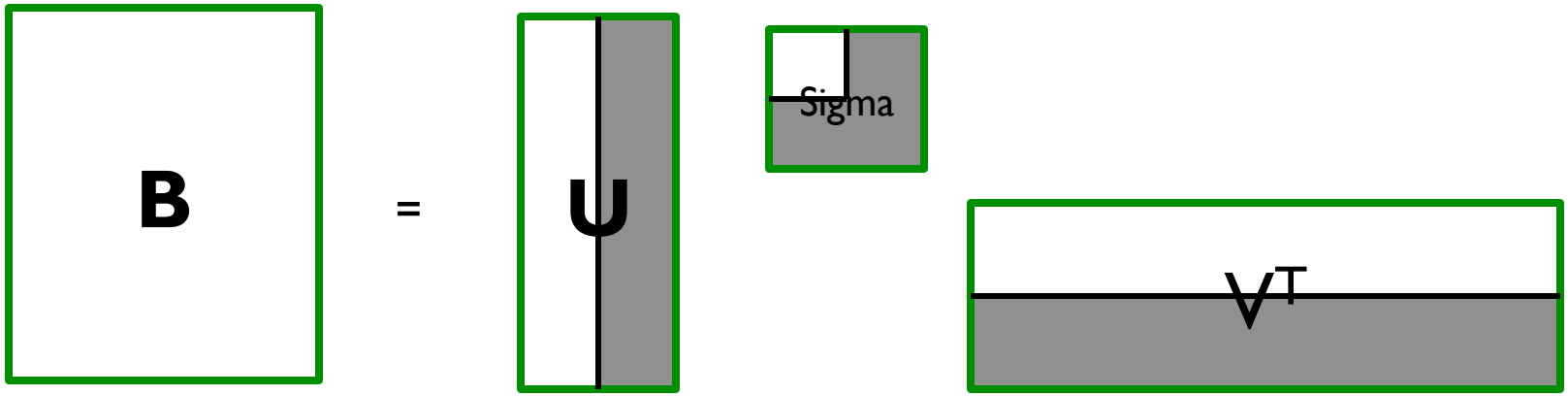
$$\|A-B\|_F = \sqrt{\Sigma_{ij} (A_{ij}-B_{ij})^2}$$

is "small"

SVD – Best Low Rank Approx.



B is best approximation of A



SVD - Interpretation #2

Equivalent:

'spectral decomposition' of the matrix:

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

SVD - Interpretation #2

Equivalent:

'spectral decomposition' of the matrix

$$\underset{n}{\overset{m}{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}}} = \overbrace{\sigma_1 \; u_1 \; v^T_1 + \sigma_2 \; u_2 \; v^T_2 + \ldots}^{k \text{ terms}}$$

$n \times 1$ $\quad$ $1 \times m$

Assume: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots \geq 0$

Why is setting small σ_i to 0 the right thing to do?

Vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are unit length, so σ_i scales them.

So, zeroing small σ_i introduces less error.

SVD - Interpretation #2

Q: How many σ_s to keep?

A: Rule-of-a thumb:
keep 80-90% of 'energy'

$$\overset{\longleftarrow\ m\ \longrightarrow}{\underset{n}{\updownarrow}} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \sigma_1 \quad u_1 \quad v^T_1 + \sigma_2 \quad u_2 \quad v^T_2 + \ldots$$

Assume: $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$

Computing SVD

(especially in a MapReduce setting…)

1. Stochastic Gradient Descent (today)
2. Stochastic SVD (not today)
3. Other methods (not today)

KDD 2011

Large-Scale Matrix Factorization with Distributed Stochastic Gradient Descent

Rainer Gemulla

talk pilfered from →

Peter J. Haas

Yannis Sismanis

Erik Nijkamp

Matrix completion for image denoising

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

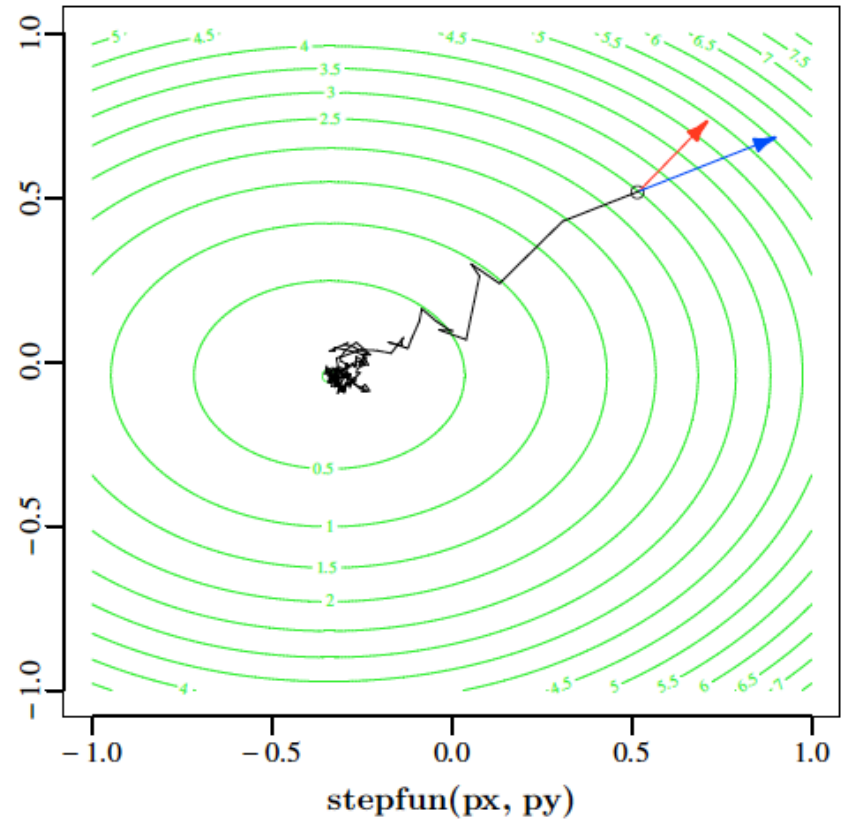
step size

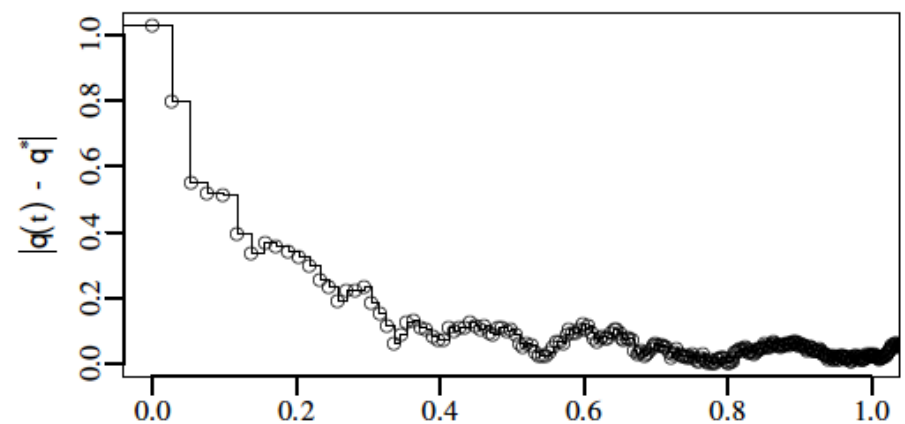
Stochastic Gradient Descent

- 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





Matrix factorization as SGD - 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.

$\boldsymbol{W}'_{i*} \leftarrow \boldsymbol{W}_{i*} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{W}_{i*}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$

$\boldsymbol{H}_{*j} \leftarrow \boldsymbol{H}_{*j} - \epsilon_n N \frac{\partial}{\partial \boldsymbol{H}_{*j}} l(\boldsymbol{V}_{ij}, \boldsymbol{W}_{i*}, \boldsymbol{H}_{*j})$

$\boldsymbol{W}_{i*} \leftarrow \boldsymbol{W}'_{i*}$

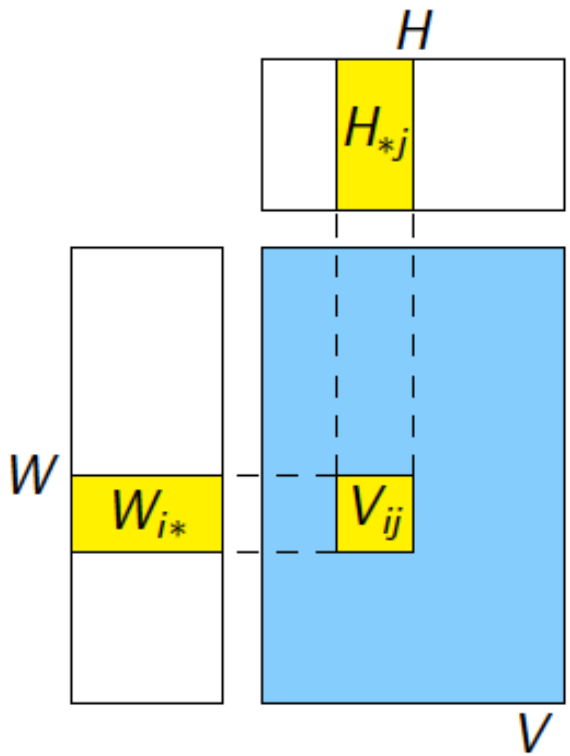
end while

step size

Matrix factorization as SGD - why does this work? Here's the 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

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

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

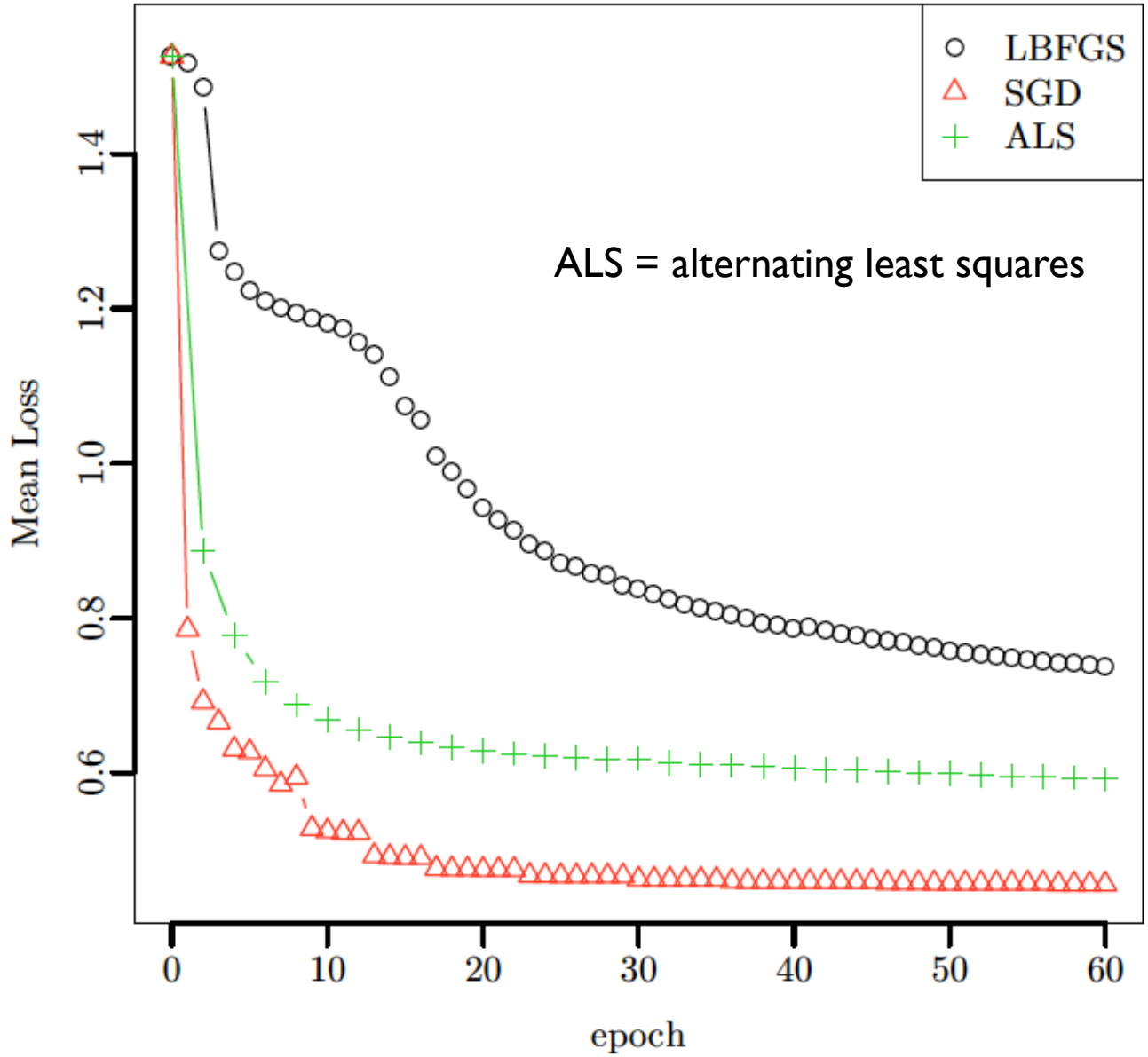
$$\frac{\partial}{\partial \boldsymbol{H}_{*j}} L(\boldsymbol{W}, \boldsymbol{H}) = \sum_{i \in Z_{*j}} \frac{\partial}{\partial \boldsymbol{W}_{*j}} L_{ij}(\boldsymbol{W}_{i*}, \boldsymbol{H}_{*j}),$$

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

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



Averaging Techniques

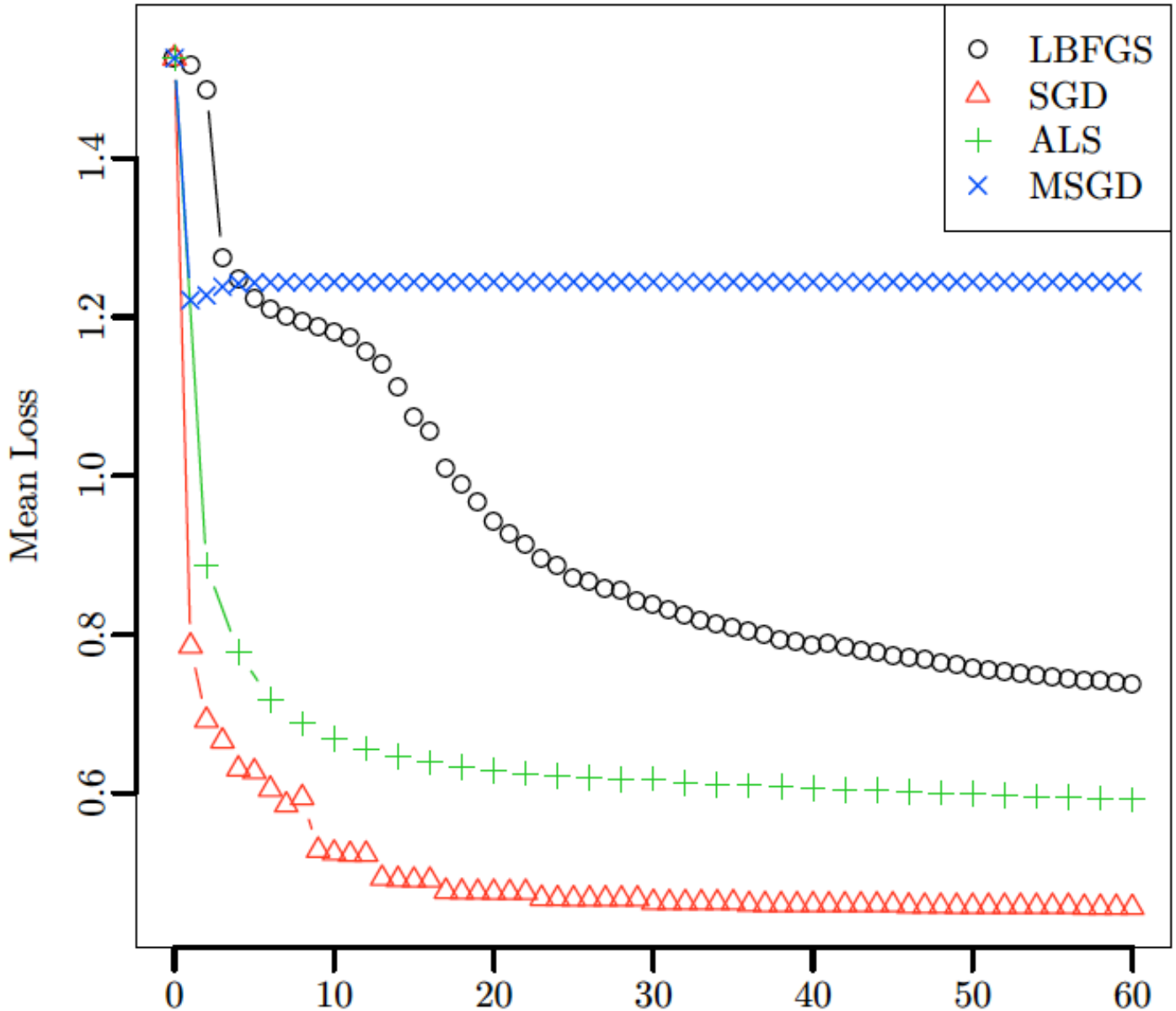
- SGD steps depend on each other

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

How to distribute?

- Parameter mixing (MSGD)
  - *Map*: Run independent instances of SGD on subsets of the data (until convergence)
  - *Reduce*: Average results

Averaging Techniques



Averaging Techniques

- SGD steps depend on each other

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

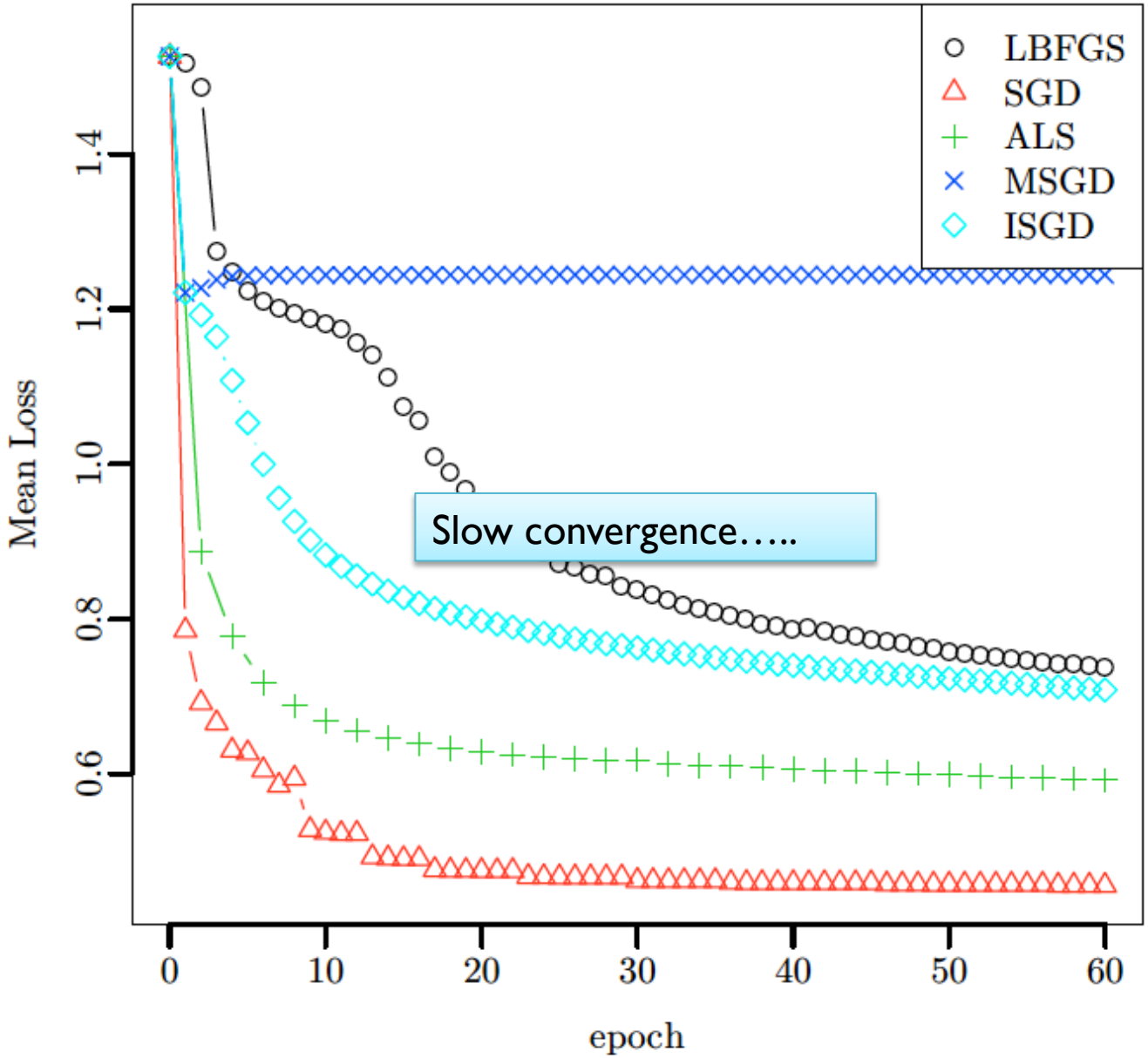
How to distribute?

- Parameter mixing (MSGD)
  - *Map*: Run independent instances of SGD on subsets of the data (until convergence)
  - *Reduce*: Average results
  - Does not converge to correct solution!

Similar to McDonnell et al with perceptron learning

- Iterative Parameter mixing (ISGD)
  - *Map*: Run independent instances of SGD on subsets of the data (for some time)
  - *Reduce*: Average results
  - Repeat

Averaging Techniques

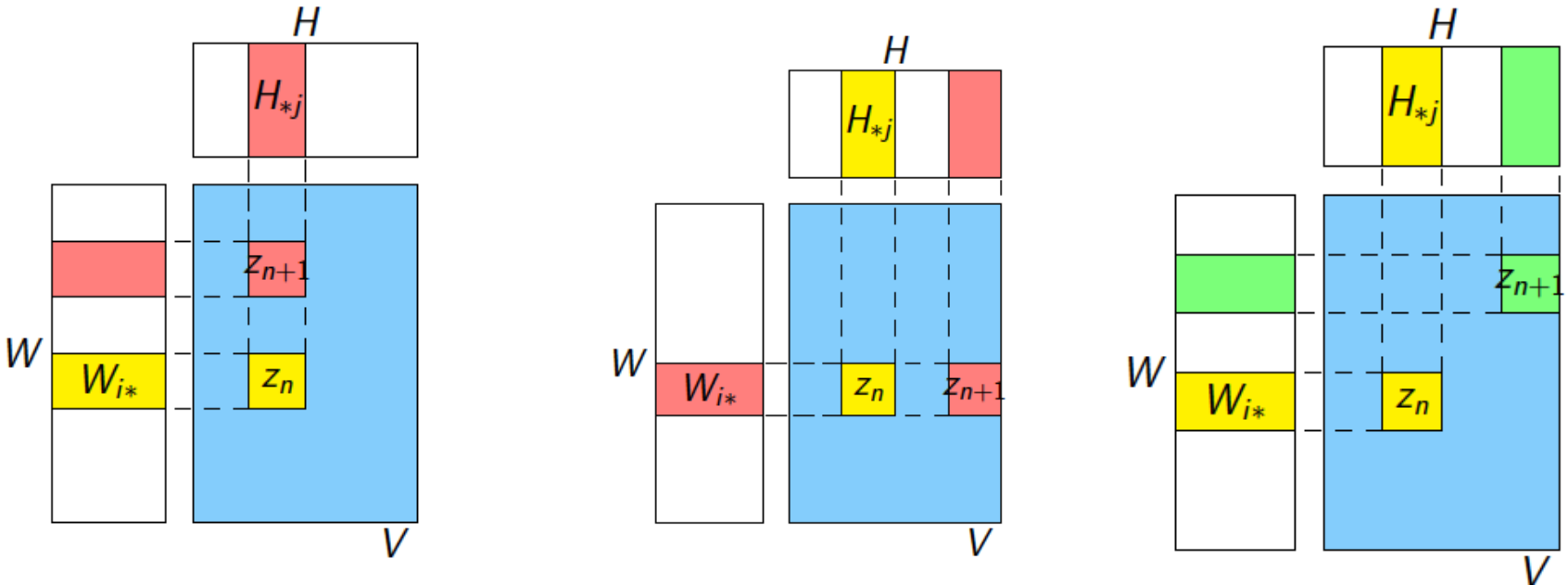


Problem Structure

- SGD steps depend on each other

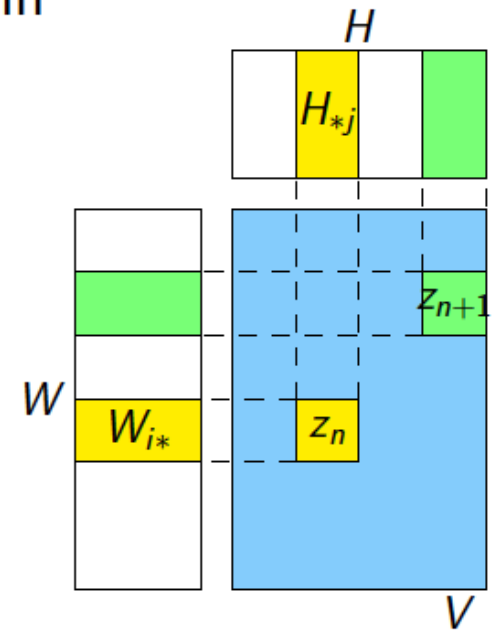
$$\theta_{n+1} = \theta_n - \epsilon_n \hat{L}'(\theta_n)$$

- An SGD step on example $z \in Z$ ...
  1. Reads $W_{i_z *}$ and $H_{* j_z}$
  2. Performs gradient computation $L'_{ij}(W_{i_z *}, H_{* j_z})$
  3. Updates $W_{i_z *}$ and $H_{* j_z}$
- Not all steps are dependent



Interchangeability

- Two elements $z_1, z_2 \in Z$ are *interchangeable* if they share neither row nor column



- When z_n and z_{n+1} are interchangeable, the SGD steps

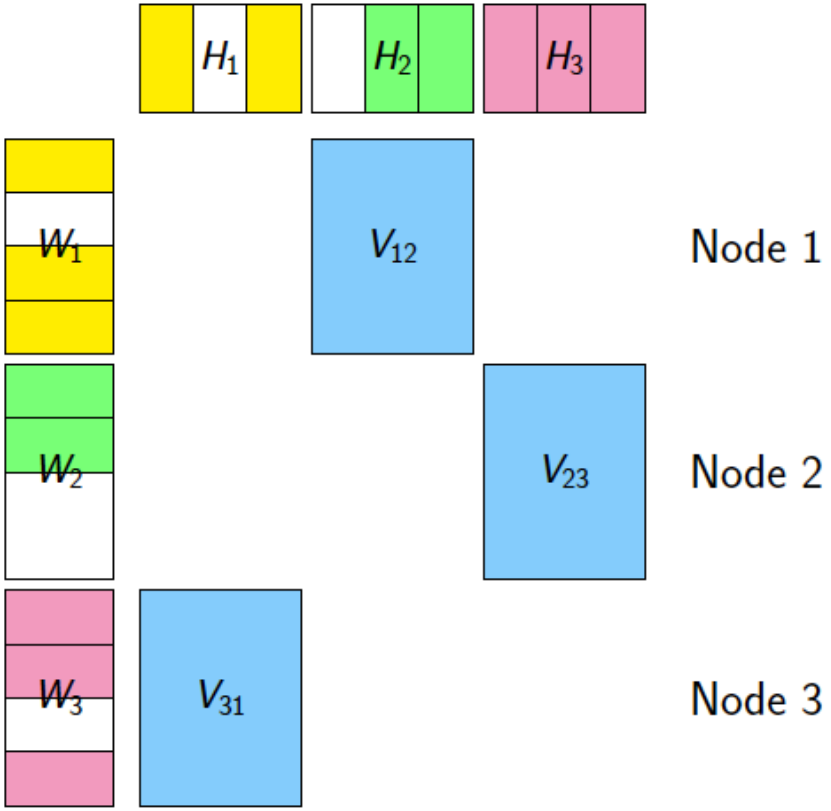
$$\begin{aligned}\theta_{n+2} &= \theta_n - \epsilon \hat{L}'(\theta_n, z_n) - \epsilon \hat{L}'(\theta_{n+1}, z_{n+1}) \\ &= \theta_n - \epsilon \hat{L}'(\theta_n, z_n) - \epsilon \hat{L}'(\theta_n, z_{n+1}),\end{aligned}$$

become parallelizable!

Exploitation

- Block and distribute the input matrix **V**
- High-level approach (Map only)
  1. Pick a "diagonal"
  2. Run SGD on the diagonal (in parallel)
  3. Merge the results
  4. Move on to next "diagonal"
  - Steps 1–3 form a *cycle*
- Step 2:
  Simulate sequential SGD
  - Interchangeable blocks
  - Throw dice of how many iterations per block
  - Throw dice of which step sizes per block

  - Instance of "stratified SGD"
  - Provably correct



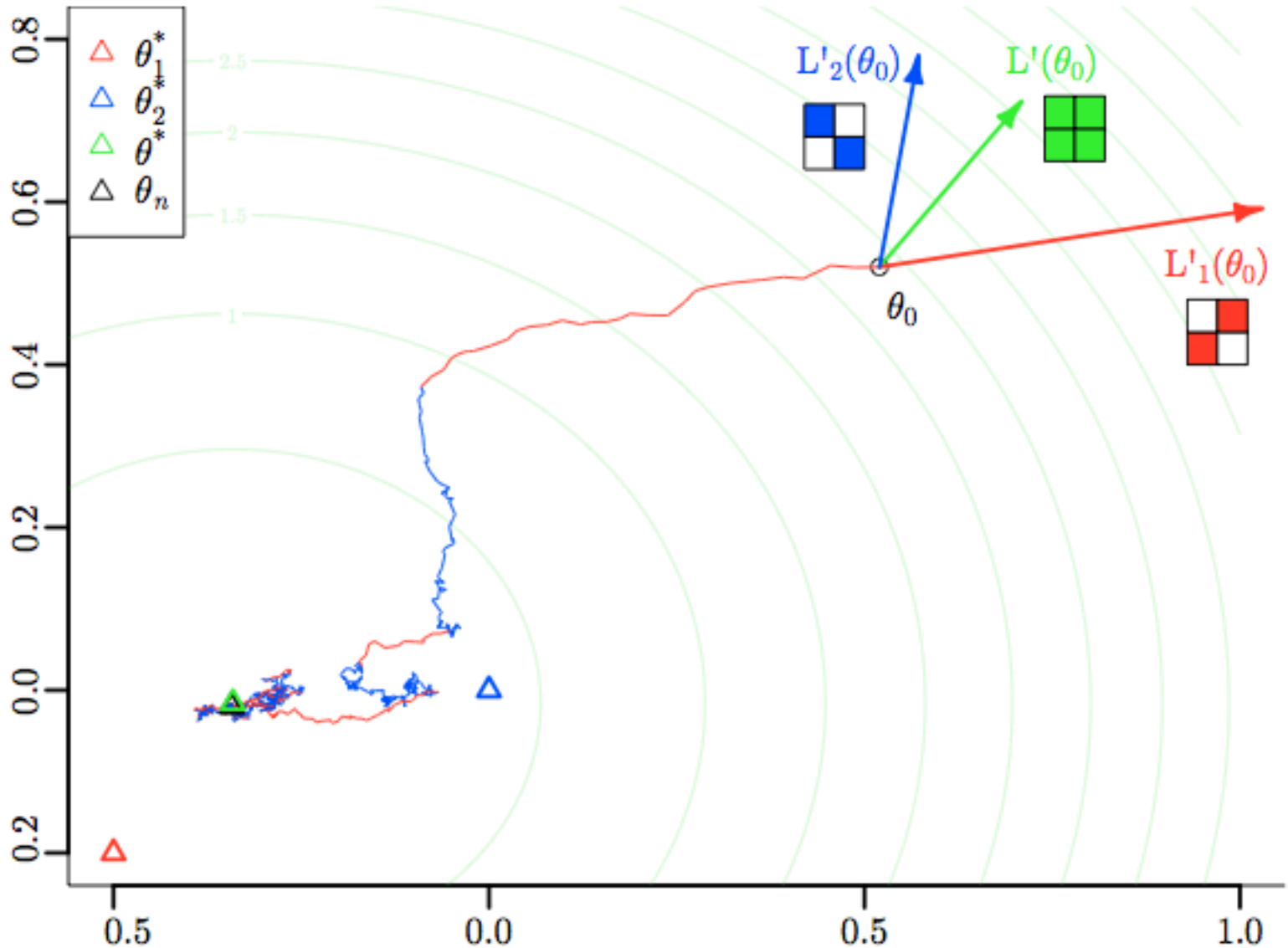


Figure 2: Example of stratified SGD

More detail….

- Randomly permute rows/cols of matrix
- Chop V,W,H into blocks of size *d x d*
  - *m/d* blocks in W, *n/d* blocks in H
- Group the data:
  - Pick a set of blocks with no overlapping rows or columns (a *stratum)*
  - Repeat until all blocks in V are covered
- Train the SGD
  - Process strata in series
  - Process blocks within a stratum in parallel

More detail....

Algorithm 2 DSGD for Matrix Factorization

Require: $\boldsymbol{Z}$, $\boldsymbol{W}_0$, $\boldsymbol{H}_0$, cluster size d

$\boldsymbol{W} \leftarrow \boldsymbol{W}_0$

$\boldsymbol{H} \leftarrow \boldsymbol{H}_0$

Block $\boldsymbol{Z}$ / $\boldsymbol{W}$ / $\boldsymbol{H}$ into $d \times d$ / $d \times 1$ / $1 \times d$ blocks

while not converged **do** /* *epoch* */

  Pick step size ϵ

  for $s = 1, \ldots, d$ **do** /* *subepoch* */

    Pick d blocks $\{\boldsymbol{Z}^{1j_1}, \ldots, \boldsymbol{Z}^{dj_d}\}$ to form a stratum

    for $b = 1, \ldots, d$ **do** /* *in parallel* */

      Run SGD on the training points in $\boldsymbol{Z}^{bj_b}$ (step size $= \epsilon$)

    end for

  end for

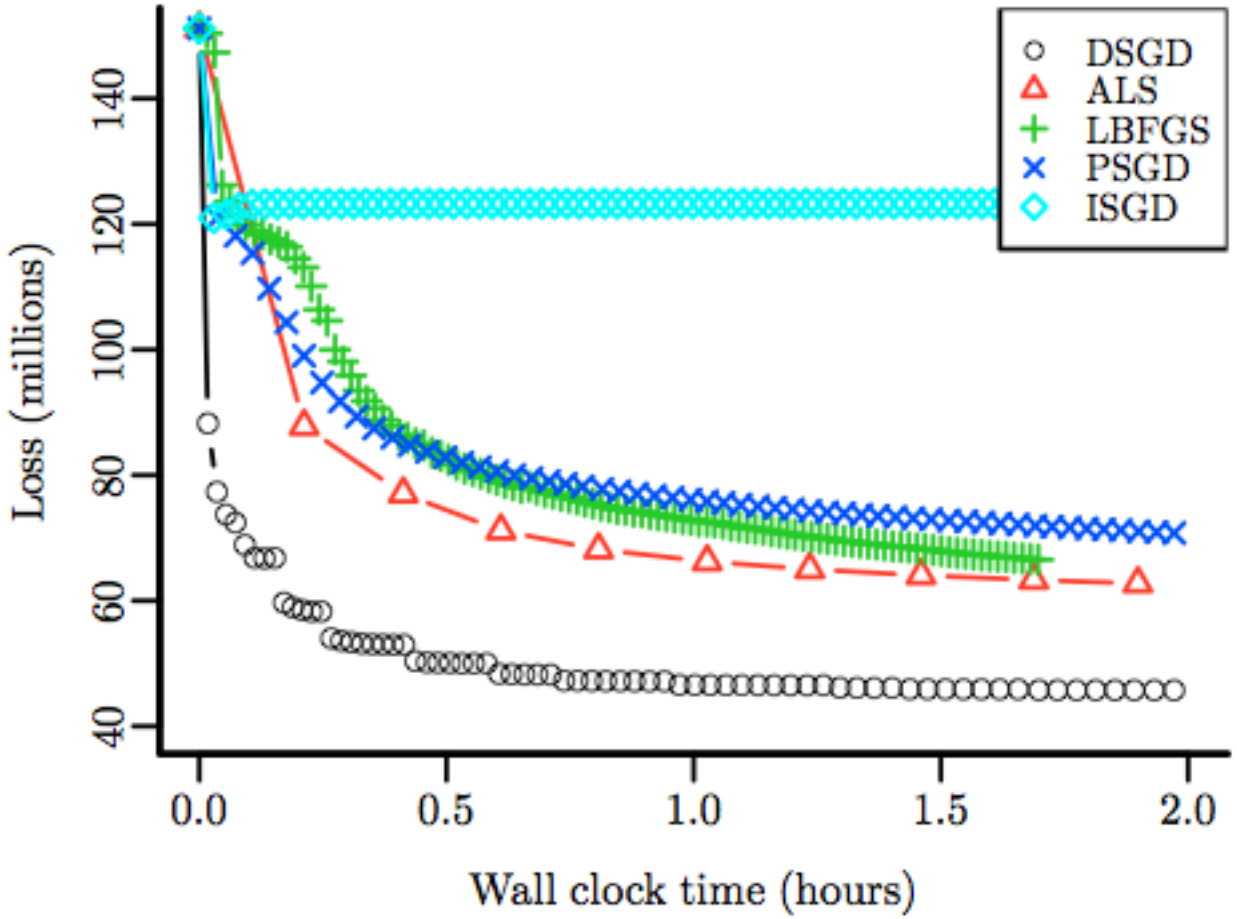
end while

Z was ***V***

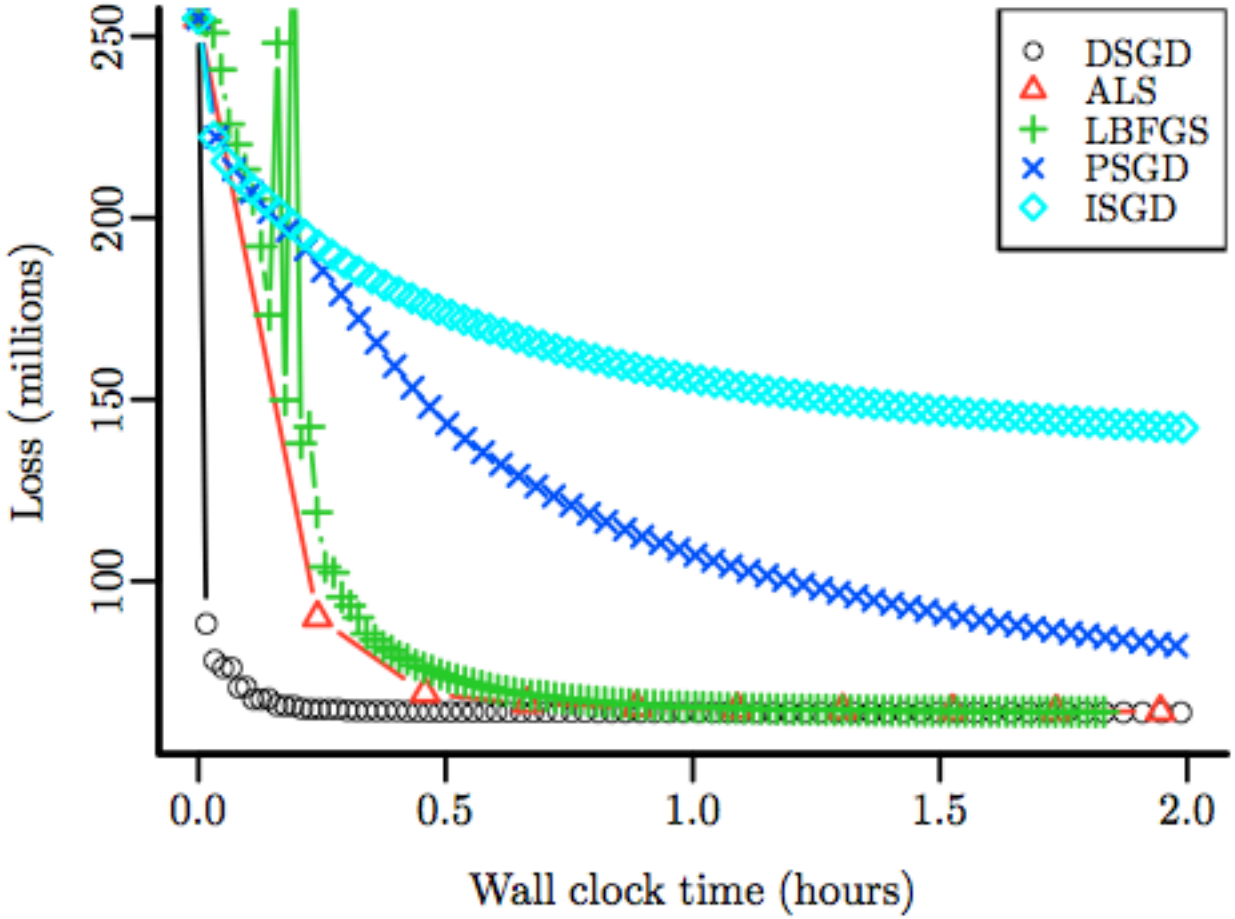
More detail….

M= $\begin{pmatrix} 1 & 2 & \cdots & d \\ 2 & 3 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ d & 1 & \cdots & d-1 \end{pmatrix}.$

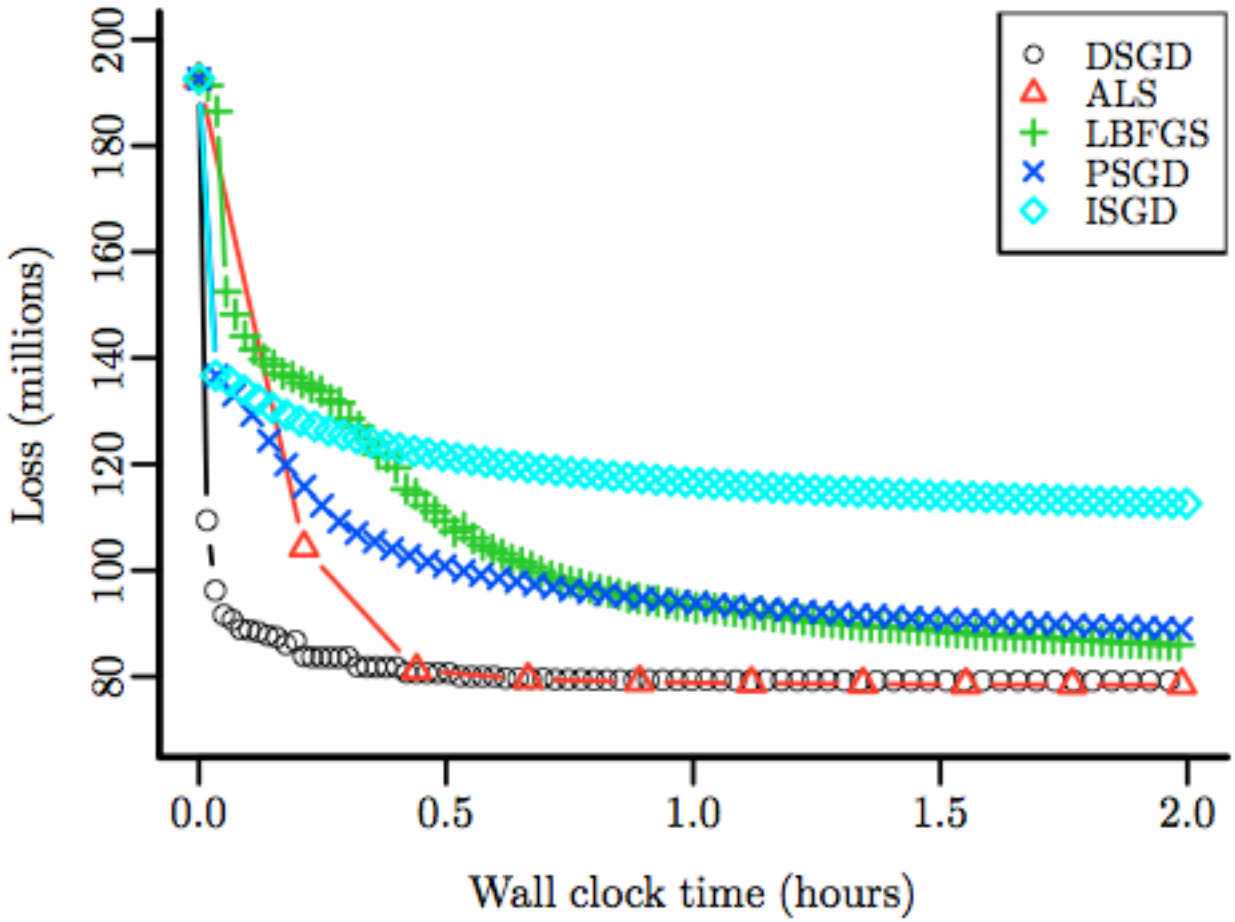
- Initialize W,H randomly
  - not at zero ☺
- Choose a random ordering (random sort) of the points in a stratum in each "sub-epoch"
- Pick strata sequence by permuting rows and columns of M, and using M'[k,i] as column index of row i in subepoch k
- Use "bold driver" to set step size:
  - increase step size when loss decreases (in an epoch)
  - decrease step size when loss increases
- Implemented in Hadoop and R/Snowfall



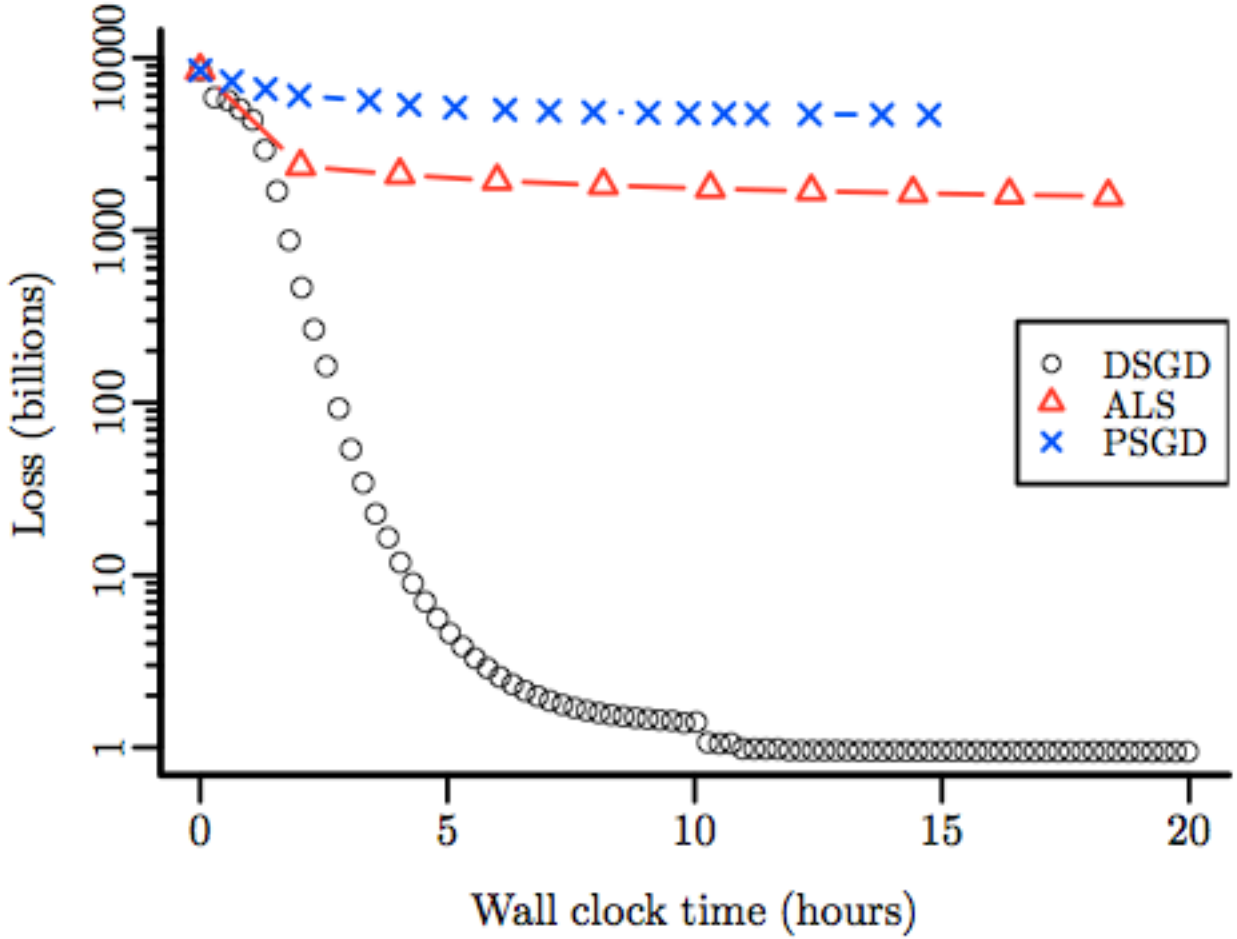
(a) Netflix, NZSL



(b) Netflix, L2, $\lambda = 50$

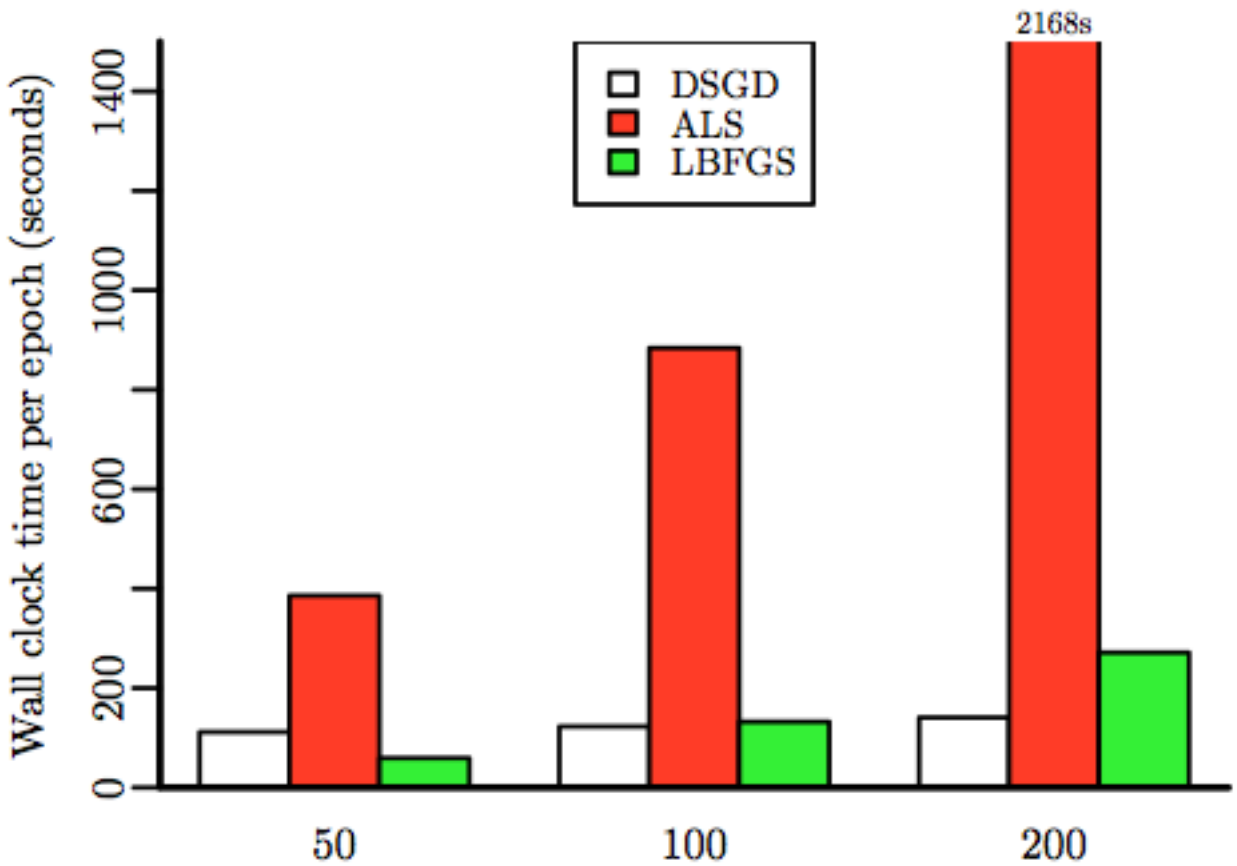


(c) Netflix, NZL2, $\lambda = 0.05$

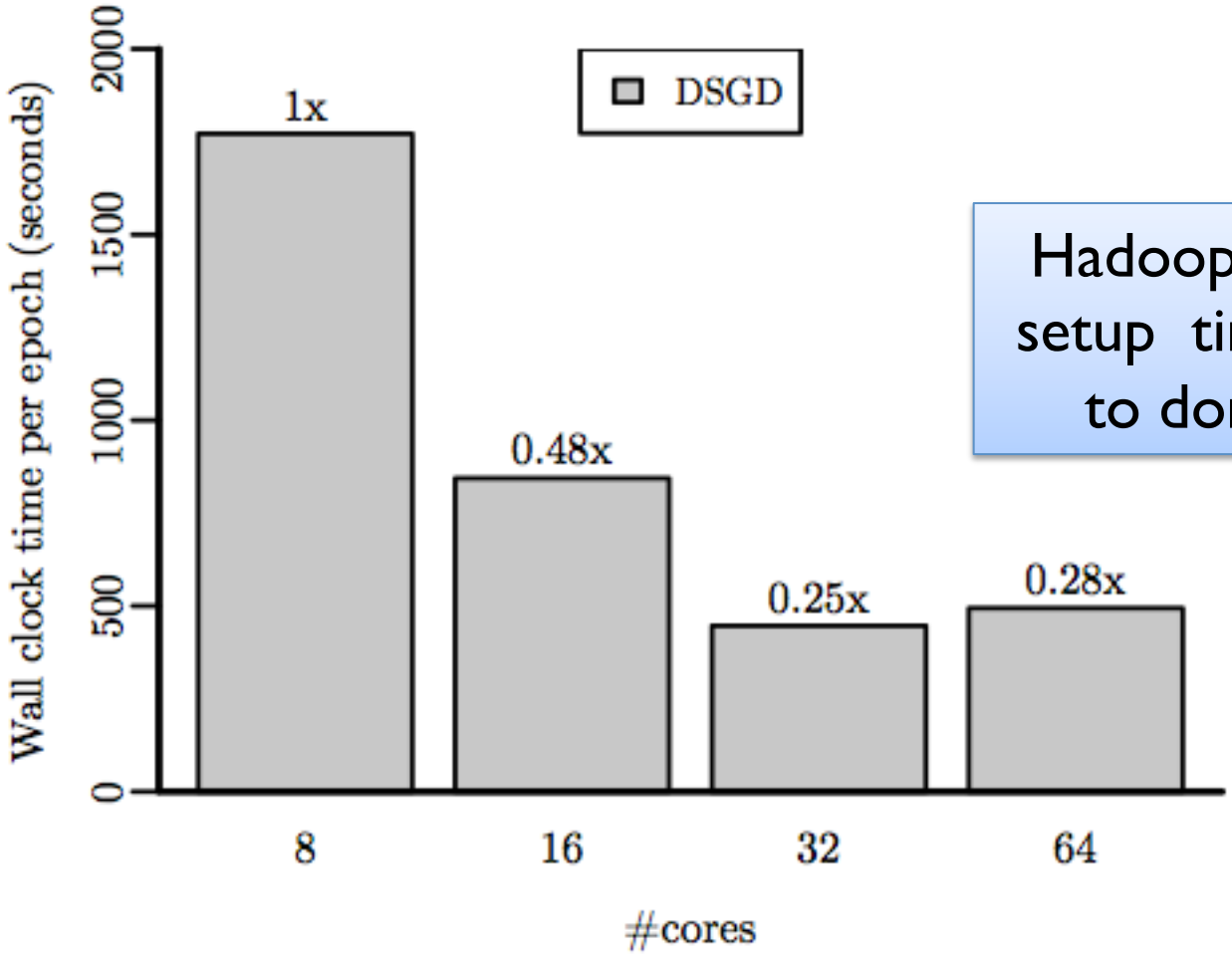


(d) Synthetic data, L2, $\lambda = 0.1$

Varying rank
100 epochs for all



Hadoop scalability



Hadoop process setup time starts to dominate

(b) Increasing cores (Hadoop, 6.4B entries)

SVD - Conclusions so far

- SVD: $\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^T$: unique
  - $\mathbf{U}$: user-to-concept similarities
  - $\mathbf{V}$: movie-to-concept similarities
  - Σ : strength of each concept

- Dimensionality reduction:
  - keep the few largest singular values (80-90% of 'energy')
  - SVD: picks up linear correlations

Relation to Eigen-decomposition

- SVD gives us:
  - $\boldsymbol{A} = \boldsymbol{U}\,\boldsymbol{\Sigma}\,\boldsymbol{V}^T$
- Eigen-decomposition:
  - $\boldsymbol{A} = \boldsymbol{X}\,\boldsymbol{\Lambda}\,\boldsymbol{X}^T$
    - A is symmetric
    - U, V, X are orthonormal ($\mathbf{U}^T\mathbf{U}=\mathbf{I}$),
    - $\boldsymbol{\Lambda}$, $\boldsymbol{\Sigma}$ are diagonal
- Now let's calculate:

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

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

Relation to Eigen-decomposition

- SVD gives us:
  - $A = U \Sigma V^T$
- Eigen-decomposition:
  - $A = X \Lambda X^T$
    - A is symmetric
    - U, V, X are orthonormal ($\mathbf{U}^T\mathbf{U}=\mathbf{I}$),
    - $\mathbf{\Lambda}, \mathbf{\Sigma}$ are diagonal
- Now let's calculate:

Shows how to compute SVD using eigenvalue decomposition!

$$\Downarrow$$

$$\underset{\downarrow}{X}\ \underset{\downarrow}{\Lambda^2}\ \underset{\downarrow}{X^T}$$

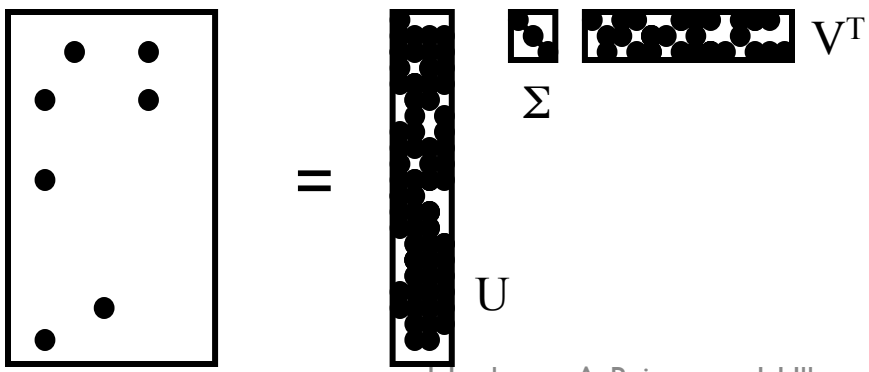
$$AA^T = U\Sigma V^T(U\Sigma V^T)^T = U\Sigma V^T(V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$$

$$A^T A = V\Sigma^T U^T(U\Sigma V^T) = \underset{\uparrow}{V}\,\underset{\uparrow}{\Sigma\Sigma^T}\,\underset{\uparrow}{V^T}$$

$$\underset{X}{\uparrow}\ \underset{\Lambda^2}{\uparrow}\ \underset{X^T}{\uparrow}$$

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!**



CUR Decomposition

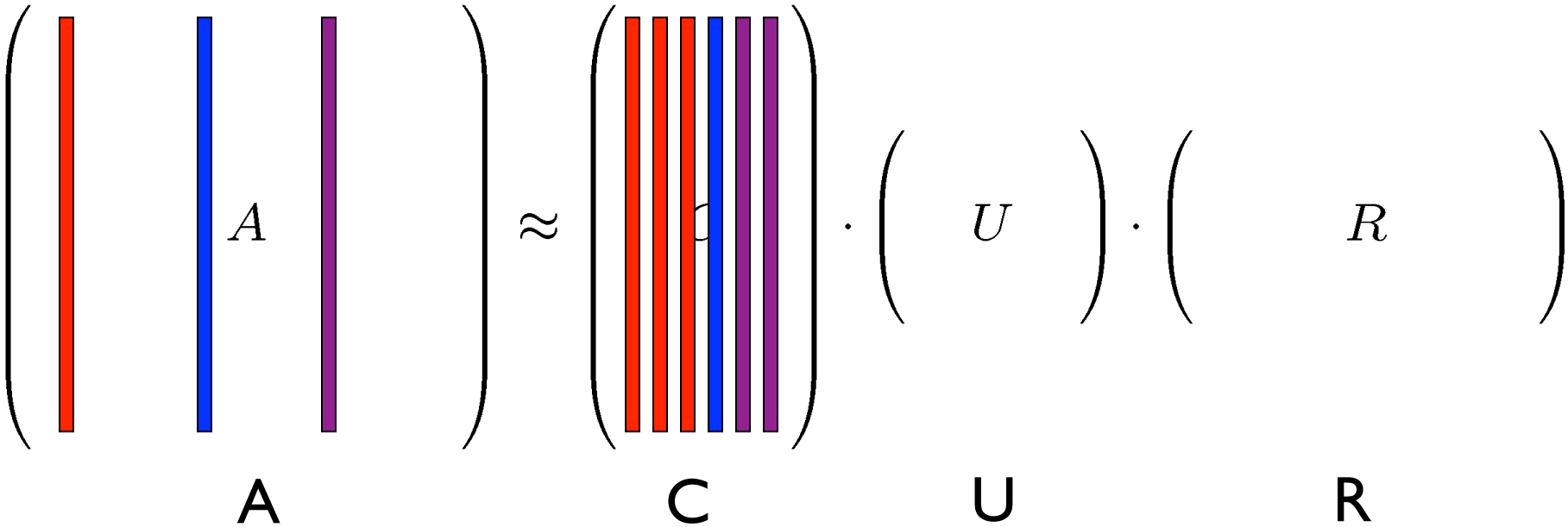
Frobenius norm:

$\|X\|_F = \sqrt{\Sigma_{ij} X_{ij}^2}$

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

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

- "Constraints" on C and R:

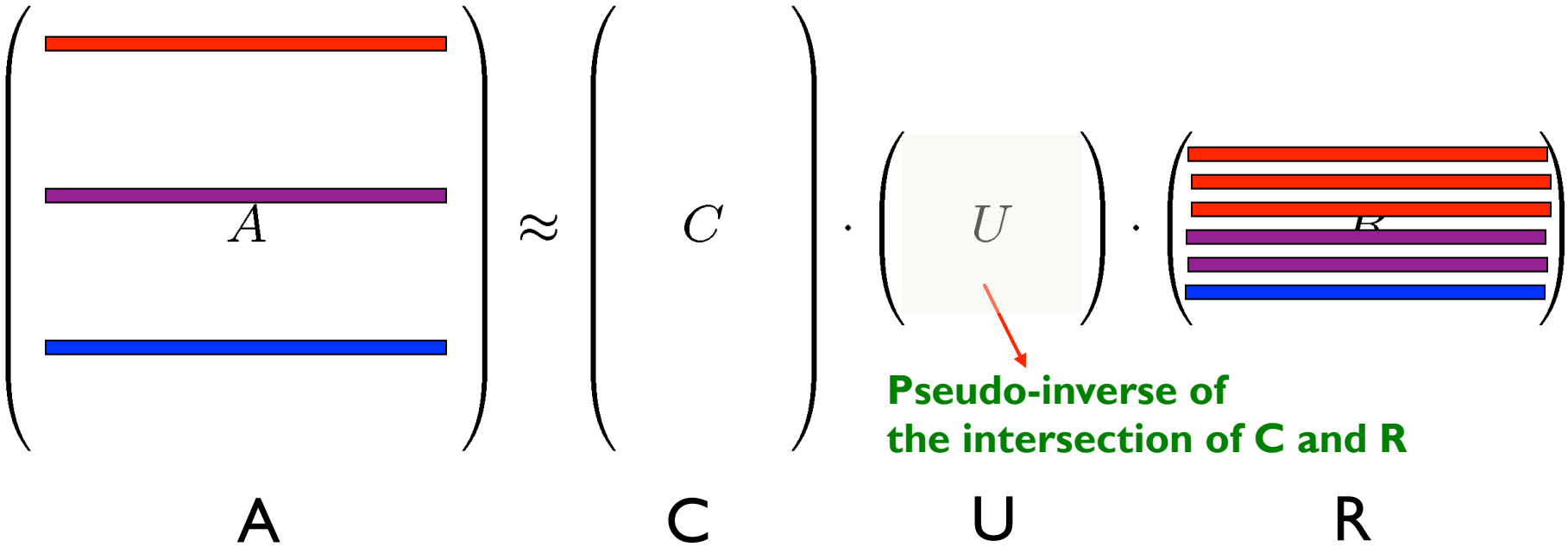


CUR Decomposition

Frobenius norm:
$\|X\|_F = \sqrt{\Sigma_{ij} X_{ij}^2}$

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

  Make $\|A-C\cdot U\cdot R\|_F$ small
- "Constraints" on C and R:



CUR: Provably good approx. to SVD

- **Let:**
  $\mathbf{A_k}$ be the "best" rank $\boldsymbol{k}$ approximation to $\mathbf{A}$ (that is, $\mathbf{A}_k$ is SVD of A)

Theorem [Drineas et al.]

CUR in $O(\mathbf{m \cdot n})$ time achieves

– $\|\mathbf{A-CUR}\|_F \leq \|\mathbf{A-A_k}\|_F + \varepsilon\|\mathbf{A}\|_F$

with probability at least $\mathbf{1-\delta}$, by picking

– $O(k \log(1/\delta)/\varepsilon^2)$ columns, and

– $O(k^2 \log^3(1/\delta)/\varepsilon^6)$ rows

In practice:
Pick 4*k* cols/rows

CUR: How it Works

- Sampling columns (similarly for rows):

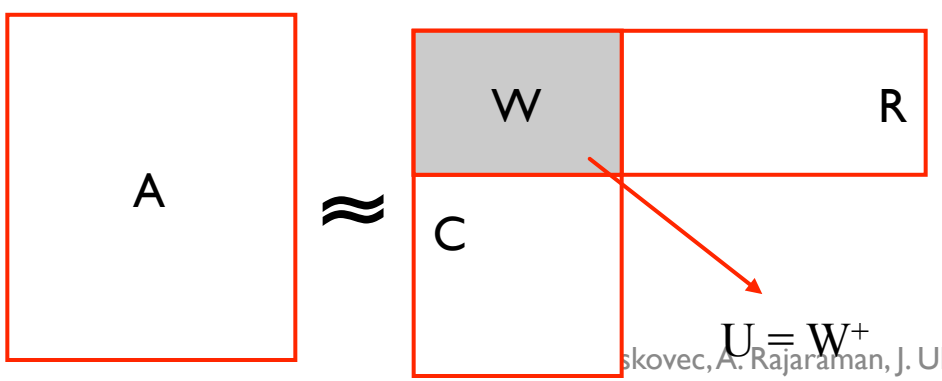
Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size c
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

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

Note this is a randomized algorithm, same column can be sampled more than once

Computing U

- Let **W** be the "intersection" of sampled columns **C** and rows **R**
  - Let SVD of $\mathbf{W} = \mathbf{X}\,\mathbf{Z}\,\mathbf{Y}^T$
- **Then:** $\mathbf{U} = \mathbf{W}^+ = \mathbf{Y}\,\mathbf{Z}^+\,\mathbf{X}^T$
  - $\mathbf{Z}^+$: **reciprocals of non-zero singular values:** $\mathbf{Z}^+_{ii} = 1/\mathbf{Z}_{ii}$
  - $\mathbf{W}^+$ is the "**pseudoinverse**"



Why pseudoinverse works?

$W = X\,Z\,Y$ then $W^{-1} = X^{-1}\,Z^{-1}\,Y^{-1}$

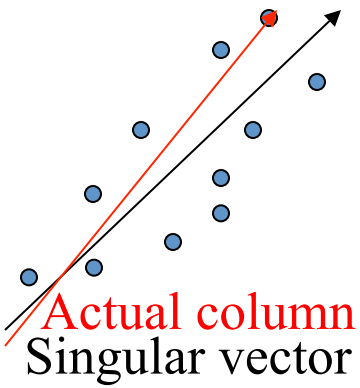
Due to orthonomality $X^{-1}=X^T$ and $Y^{-1}=Y^T$

Since Z is diagonal $Z^{-1} = 1/Z_{ii}$

Thus, if **W** is nonsingular, pseudoinverse is the true inverse

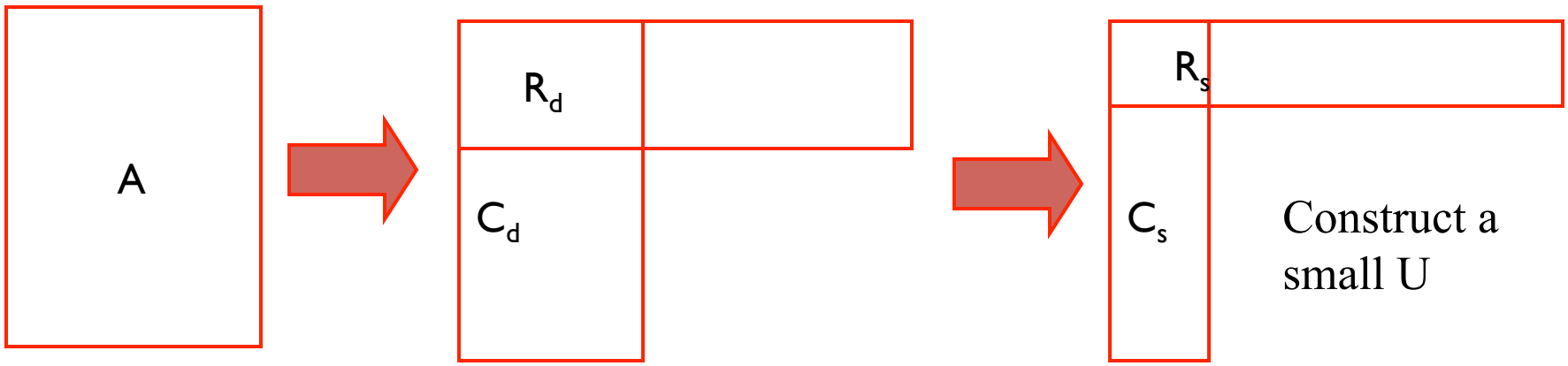
CUR: Pros & Cons

+ 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



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



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

Stochastic SVD

- “Randomized methods for computing low-rank approximations of matrices”, Nathan Halko 2012
- “Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions”, Halko, Martinsson, and Tropp, 2011
- “An Algorithm for the Principal Component Analysis of Large Data Sets”, Halko, Martinsson, Shkolnisky, and Tygert, 2010