Introduction to Data Mining

Lecture #22: Dimensionality Reduction-2

U Kang
Seoul National University

In This Lecture

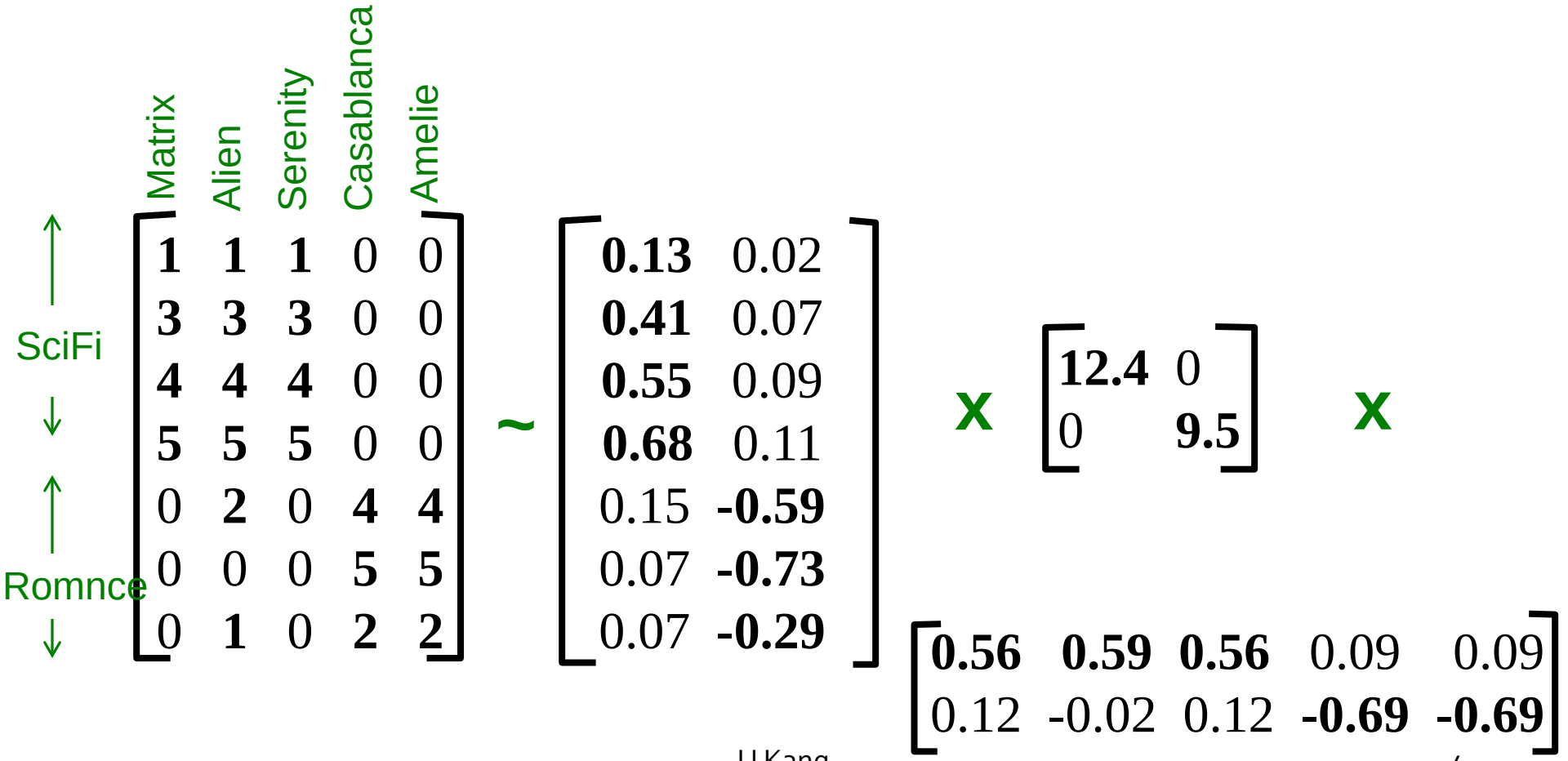
- Learn how to answer queries using SVD
- Learn the motivation and definition of CUR decomposition, an alternative method for SVD
- Compare CUR and SVD

Outline

- **SVD Case Studies**
- CUR Decomposition

Case study: How to query?

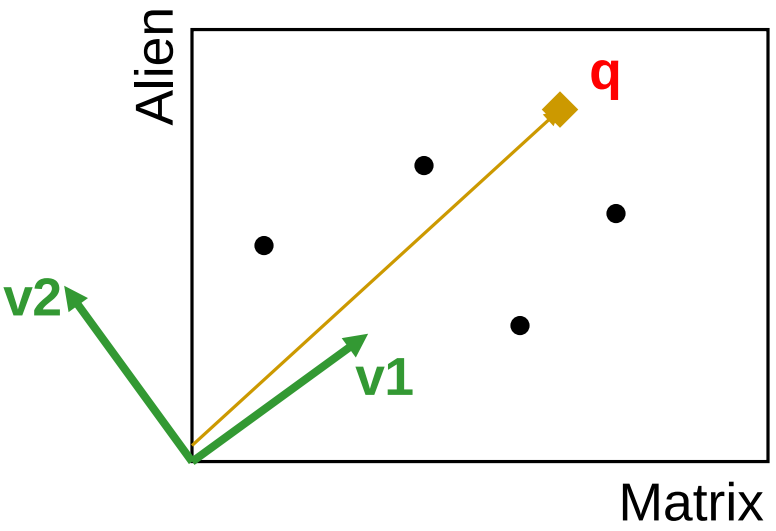
- Q: Find users that like 'Matrix'
- A: Map query into a 'concept space' – how?



Case study: How to query?

- **Q: Find users that like 'Matrix'**
- **A: Map query into a 'concept space' – how?**

$$q = \begin{bmatrix} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ 5 & 0 & 0 & 0 & 0 \end{bmatrix}$$



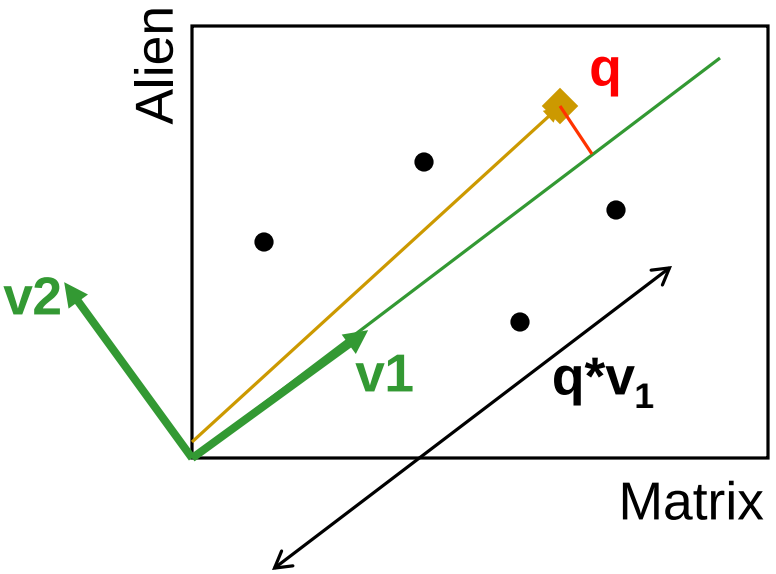
Project into concept space:
Inner product with each 'concept' vector $\mathbf{v}_i$

Case study: How to query?

- **Q: Find users that like 'Matrix'**
- **A: Map query into a 'concept space' – how?**

	Matrix	Alien	Serenity	Casablanca	Amelie
q =	5	0	0	0	0

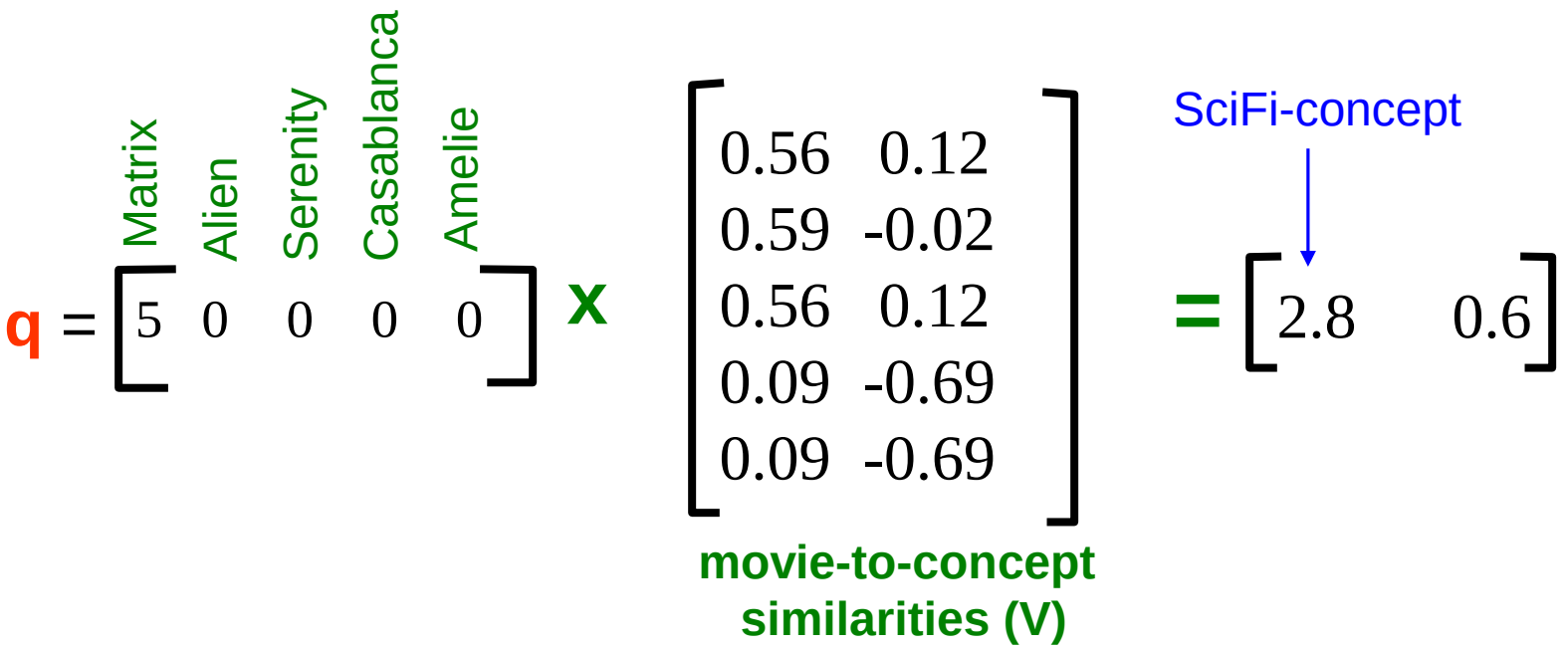
Project into concept space:
Inner product with each 'concept' vector $\mathbf{v_i}$



Case study: How to query?

Compactly, we have:

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



Case study: How to query?

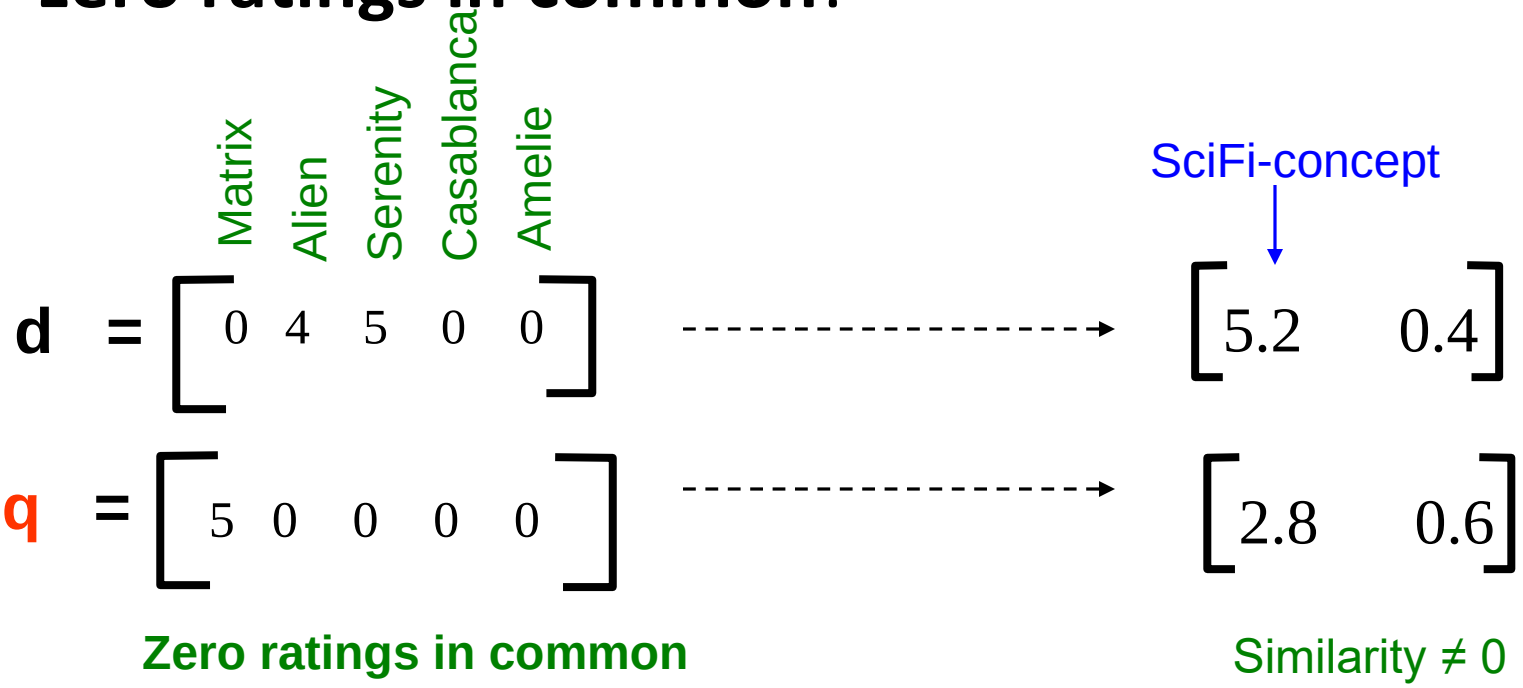
- **How would the user *d* that rated ('Alien', 'Serenity') be handled?**
  $d_{concept} = d\ V$

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

movie-to-concept similarities (V)

Case study: How to query?

- **Observation:** User ***d*** that rated (‘*Alien*’, ‘*Serenity*’) will be **similar** to user **q** that rated (‘*Matrix*’), although ***d*** and **q** have **zero ratings in common**!

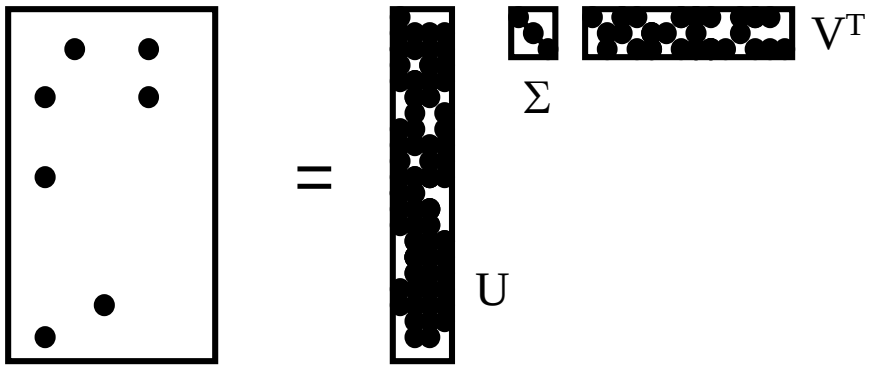


SVD: Pros and Cons

+ **Optimal low-rank approximation** in terms of Frobenius norm (or Euclidean distance)
- **Interpretability problem:**
  - A singular vector specifies a linear combination of all input columns or rows
- **Lack of sparsity:**
  - Singular vectors are **dense!**

Frobenius norm:

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



Outline

- ☑ SVD Case Studies
- ☐ **CUR Decomposition**

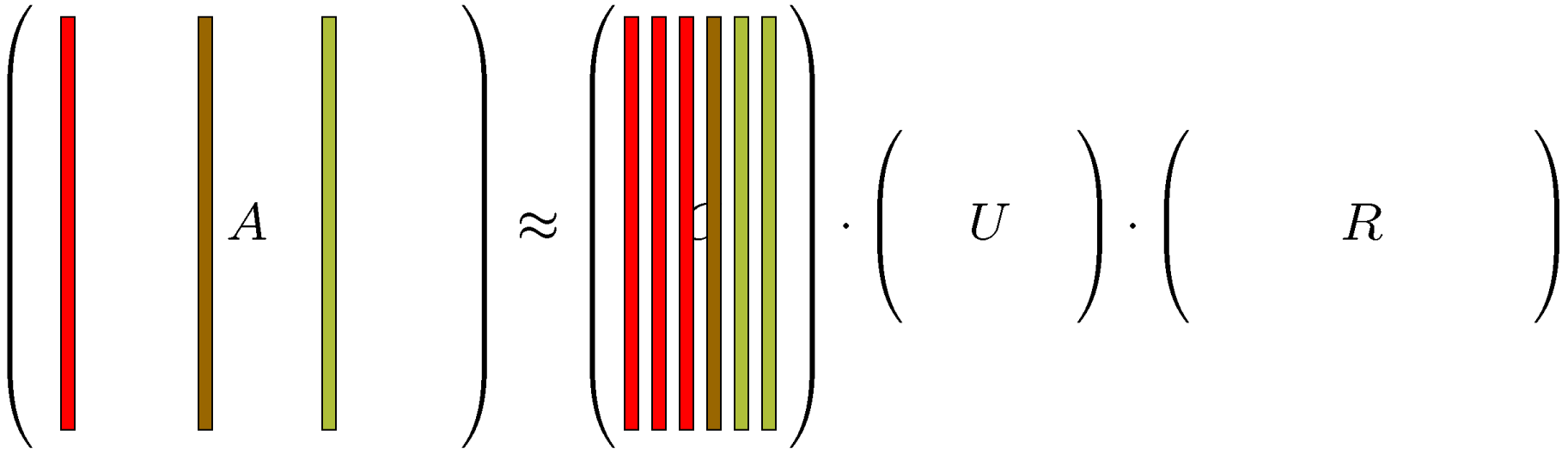
CUR Decompositior

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:**



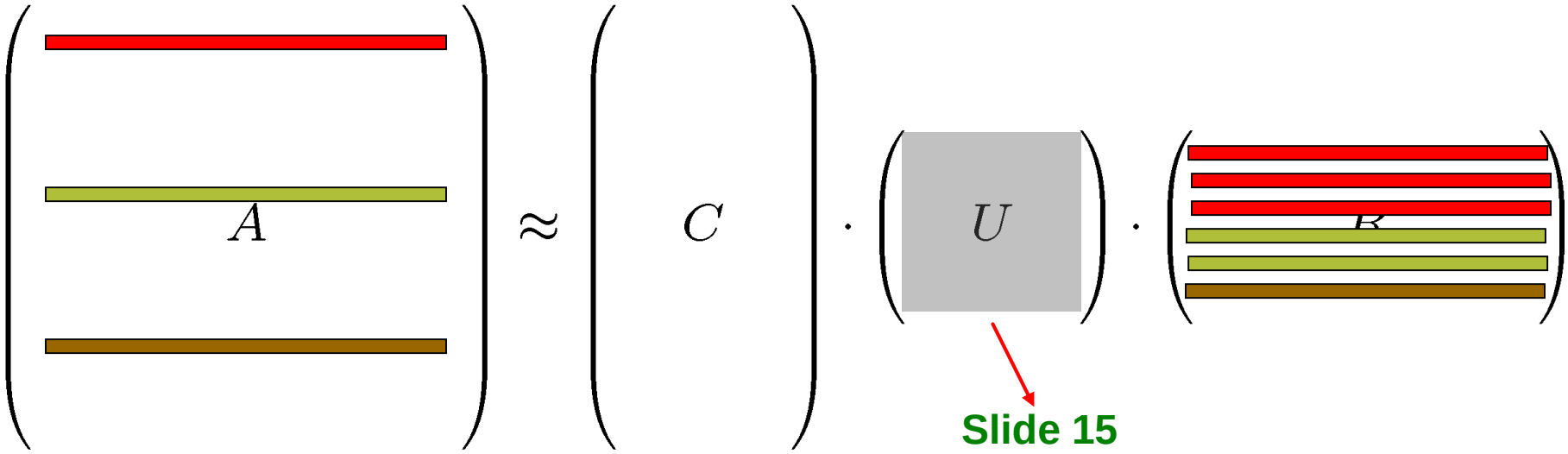
CUR Decompositior

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:**



Slide 15

CUR: How it Works

- **Sampling columns (similarly for rows):**

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

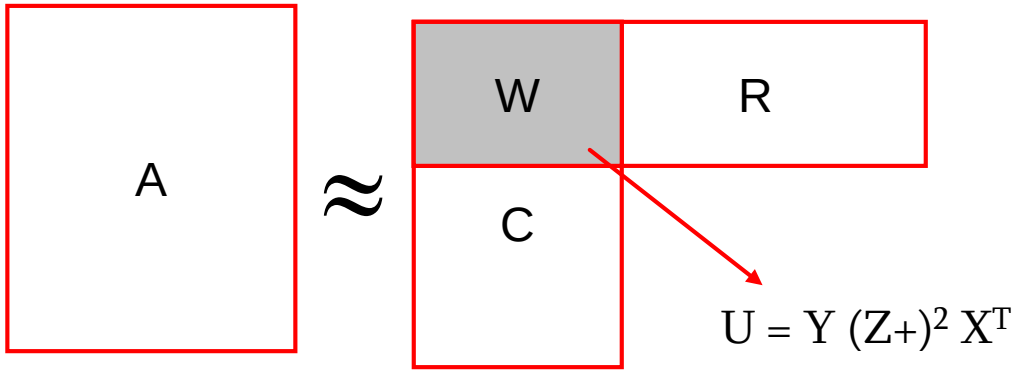
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

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 **W** = **X Z** $\mathbf{Y}^T$
- **Then: U = Y** $(\mathbf{Z}^+)^2$ $\mathbf{X}^T$
  - Z^+: **reciprocals of non-zero singular values:** $Z^+_{ii} = 1/Z_{ii}$
  - Z^+ is called "**pseudoinverse**" of Z



CUR: Provably good approx. to SVD

- **If we carefully choose # of columns and rows,**

SVD error

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

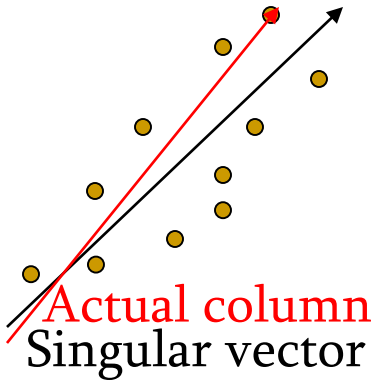
CUR error

- with probability 98%

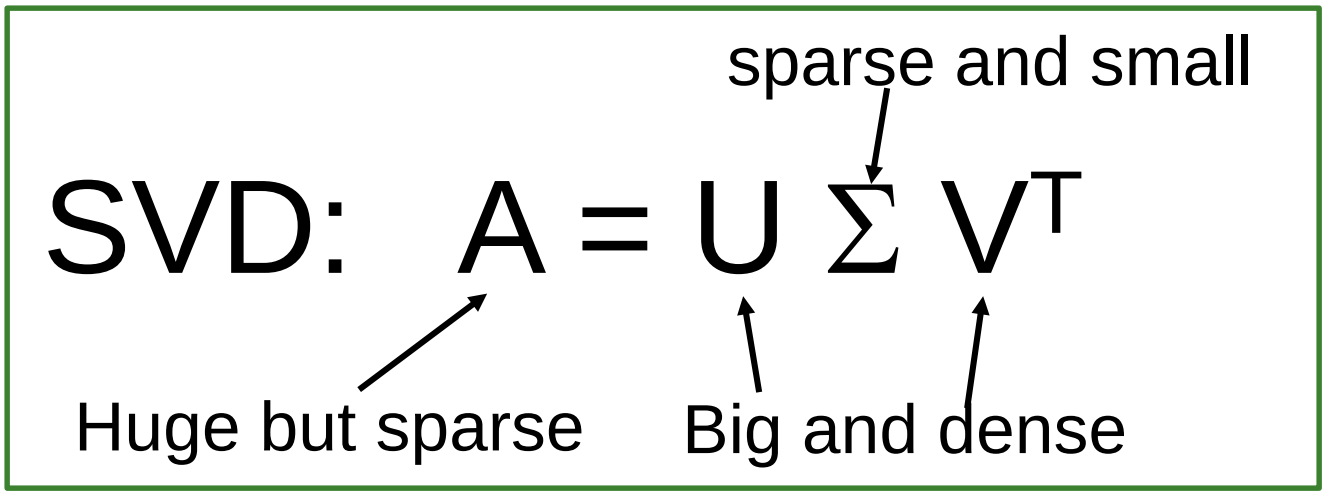
In practice:
Pick 4*k* cols/rows
for a "rank-k" approximation

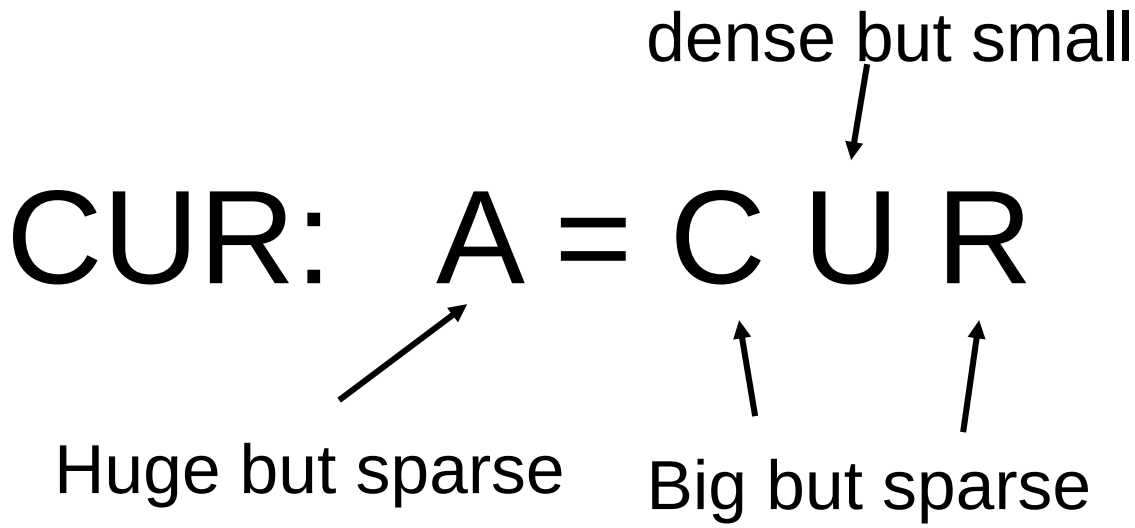
CUR: Pros & Cons

+ **Easy interpretation**
  - Since the basis vectors are actual columns and rows
+ **Sparse basis**
  - Since the basis vectors are actual columns and rows
- **Not the 'optimal' solution for minimizing error**
- **Duplicate columns and rows**
  - Columns of large norms will be sampled many times
    - CMD method (Sun et al., 2007) solves the problem



SVD vs. CUR

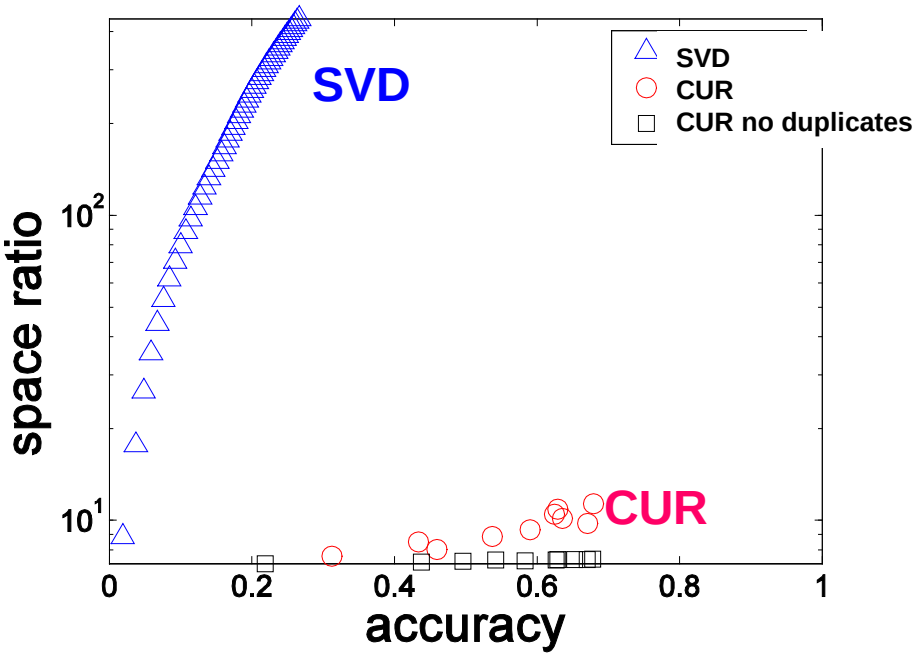

sparse and small
SVD: A = U Σ V^T
Huge but sparse
Big and dense

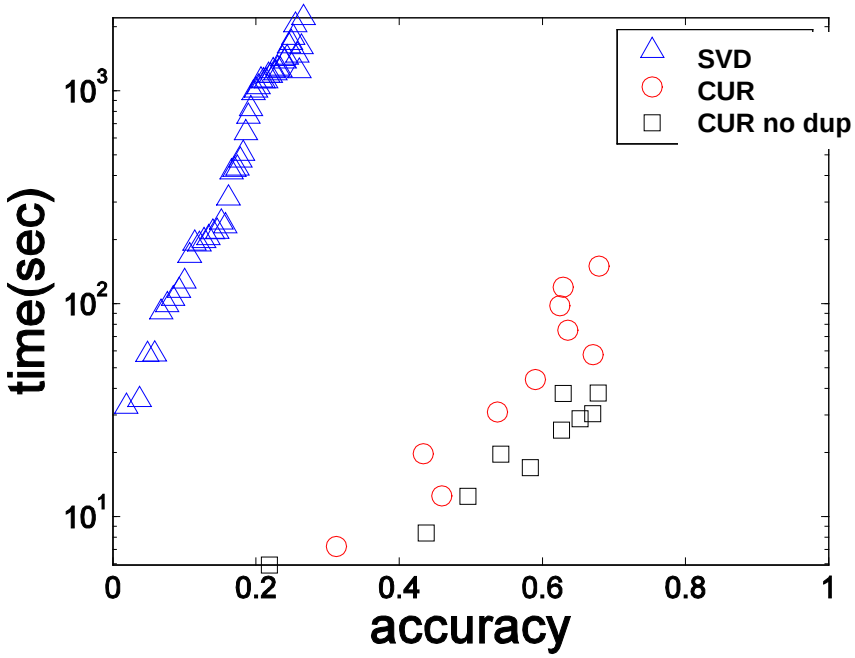

dense but small
CUR: A = C U R
Huge but sparse
Big but sparse

SVD vs. CUR: Simple Experiment

- **DBLP bibliographic data**
  - Author-to-conference big sparse matrix
  - A_{ij}: Number of papers published by author *i* at conference *j*
  - 428K authors (rows), 3659 conferences (columns)
    - **Very sparse**
- **Want to reduce dimensionality**
  - How much time does it take?
  - What is the reconstruction error?
  - How much space do we need?

Results: DBLP- big sparse matrix





- **Accuracy:**
  - 1 – relative sum squared errors
- **Space ratio:**
  - #output matrix entries / #input matrix entries
- **CPU time**

Sun, Faloutsos: *Less is More: Compact Matrix Decomposition for Large Sparse Graphs*, SDM '07.

Further Reading: 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)

Conclusion

Data contains value and knowledge

What did we learn?

- **We learned to mine different types of data:**
  - Data is high dimensional
  - Data is a graph
  - Data is infinite/never-ending

- **We learned to use different models of computation:**
  - MapReduce
  - Streams and online algorithms
  - Single machine in-memory

What did we learn?

- **We learned to solve real-world problems:**
  - Recommender systems
  - Market basket analysis
  - Spam detection
  - Duplicate document detection
  - Ranking in graphs
  - Community detection
  - Increasing the revenue of search engines
  - Trending topics in social network news feeds

What did we learn?

- **We learned various "tools":**
  - Linear algebra (SVD, Rec. Sys., Communities)
  - Pruning technique (frequent itemsets)
  - Hashing (LSH, Bloom filters)
  - Link Analysis (PageRank, HITS, Random Walk)
  - Graph Algorithms (betweenness, BFS)
  - Clustering (k-means, CURE)
  - Sampling (reservoir sampling)
  - Greedy Algorithm (Balance)

Final Remark

- This course covered extensive materials; thanks for all the hard works (quizzes, homeworks, exams, questions, etc.)
- The knowledge you learned from this course would be helpful in many ways
  - E.g., which subjects to study more; understanding and evaluating other people's work
- Takeaway points
  - Summarize each topic using few sentences so that you can really use them (e.g., Bloom Filter)
    - 1) What is the problem to solve? 2) What is the main idea? 3) Result?
  - Learn the way how each problem is solved
    - To apply the similar techniques to solve other problems in the future

Thank You!