Distributed CUR Decomposition for Bi-Clustering

Stephen Kline, Kevin Shaw

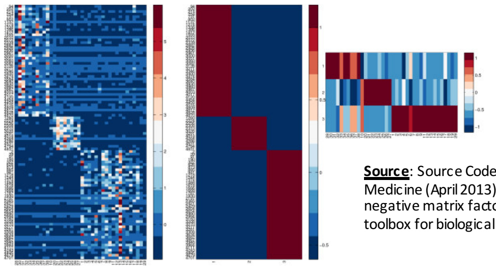
June 1, 2016

{sakline, keshaw}@stanford.edu

Stanford University, CME 323 Final Project

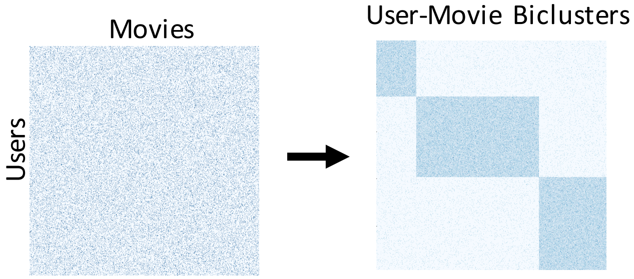
CUR as alternative to SVD – e.g. Biclustering

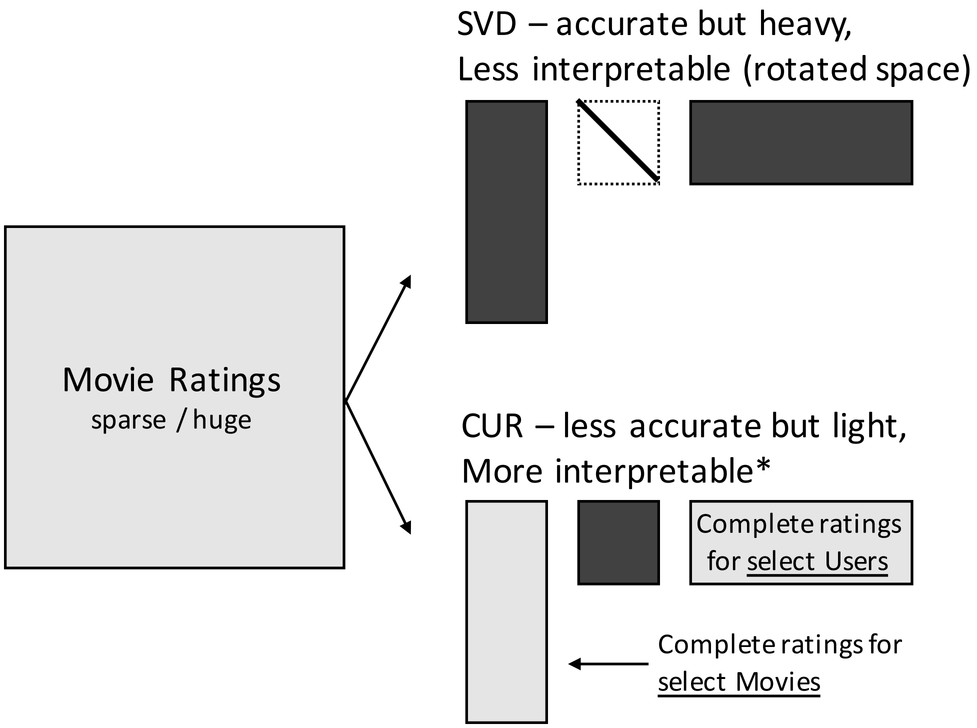
Biclustering was originally developed in the context of DNA microarrays



Source: Source Code for Biology Medicine (April 2013) - "The non-negative matrix factorization toolbox for biological data mining"

Biclustering also has potential in other areas and has added interpretability





SVD – accurate but heavy,
Less interpretable (rotated space)

Movie Ratings
sparse / huge

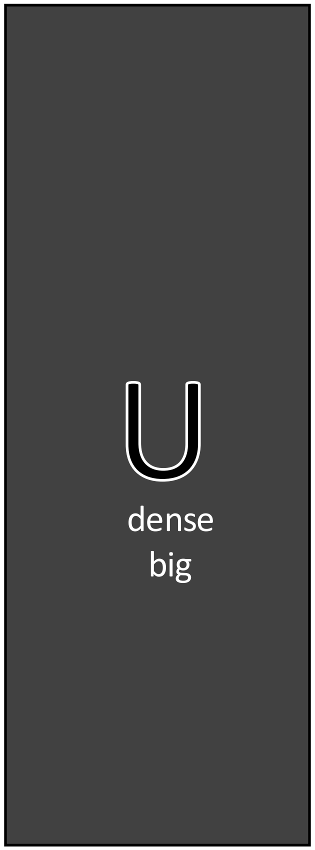
CUR – less accurate but light,
More interpretable*

Complete ratings for select Users

Complete ratings for select Movies

* As archetypal users and movies

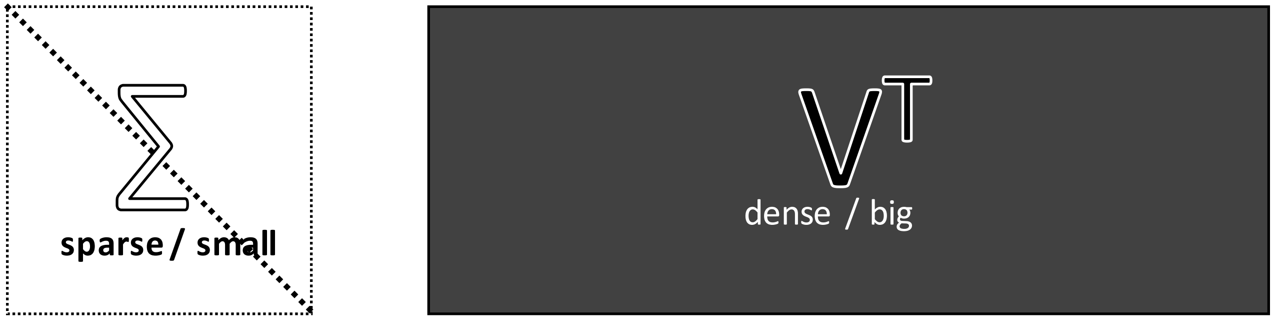
Review of SVD: $A = U\Sigma V^T$



U

dense

big



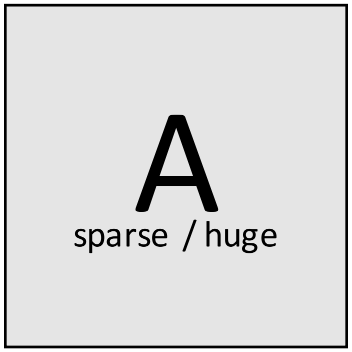
Σ

sparse / small

V^T

dense / big

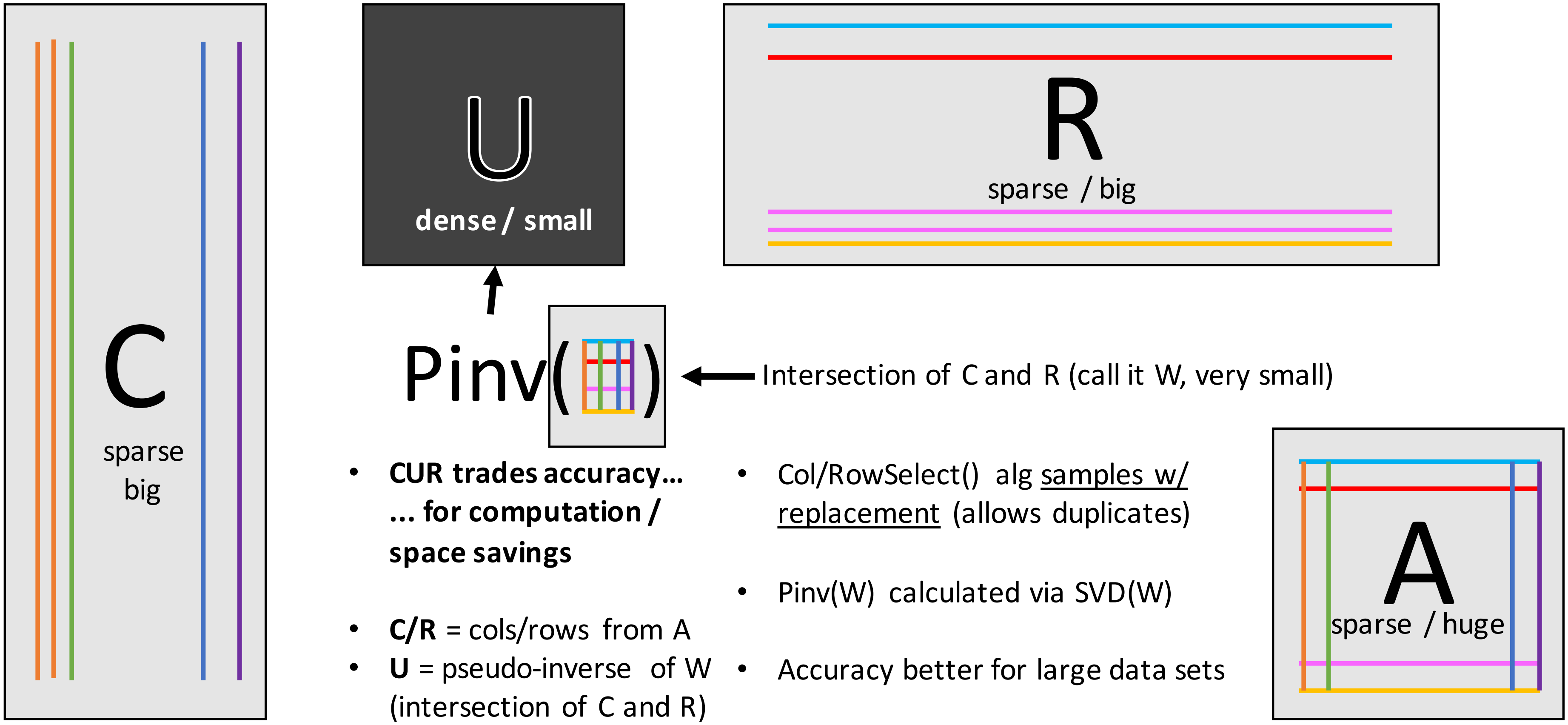
- **PRO - High accuracy**
  - k singular values/vectors produce the best k-rank approximation to A
- **CON - High computation / space requirements**
  - In our biclustering application with MovieLens data, the distributed SVD is “roughly square” - ARPACK (vs. “tall and skinny” – A^TA trick)



A

sparse / huge

Background on A = CUR



Intersection of C and R (call it W, very small)

- **CUR trades accuracy…**
  … for computation / space savings

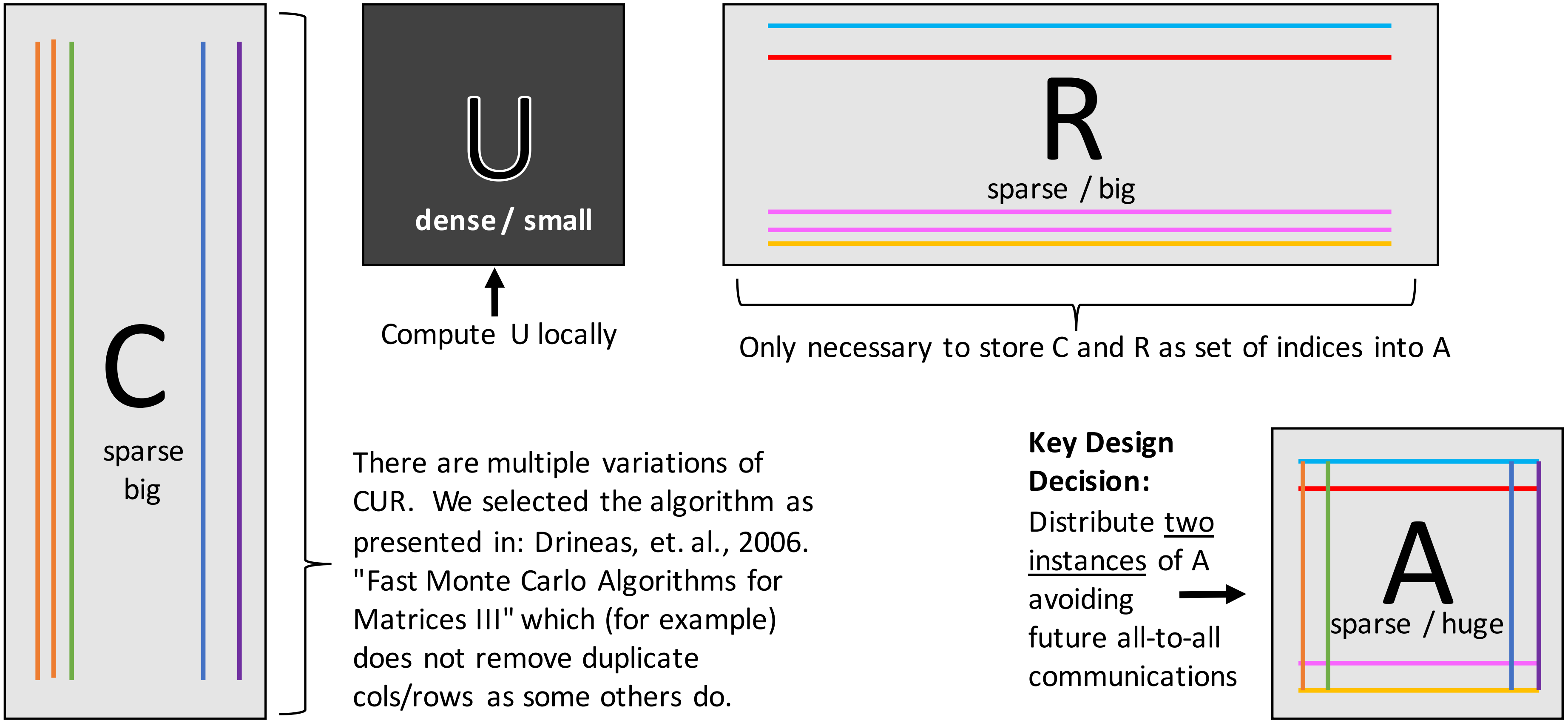
- **C/R** = cols/rows from A
- **U** = pseudo-inverse of W (intersection of C and R)

- Col/RowSelect() alg samples w/ replacement (allows duplicates)

- Pinv(W) calculated via SVD(W)

- Accuracy better for large data sets

Design Decisions for Distributed CUR



C

sparse

big

U

dense / small

Compute U locally

R

sparse / big

Only necessary to store C and R as set of indices into A

There are multiple variations of CUR. We selected the algorithm as presented in: Drineas, et. al., 2006. "Fast Monte Carlo Algorithms for Matrices III" which (for example) does not remove duplicate cols/rows as some others do.

Key Design Decision:

Distribute two instances of A avoiding future all-to-all communications

A

sparse / huge

Serial vs. Distributed CUR - Asymptotics

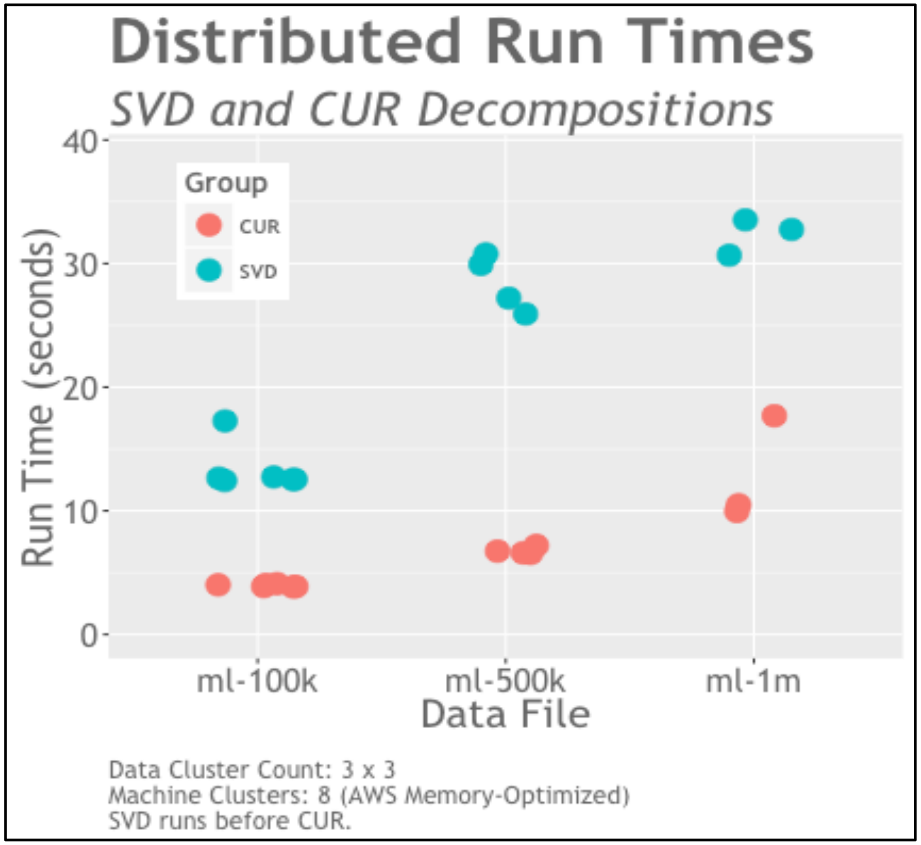
Serial

- Build C and R:
  - **Generate probabilities – O(mn)**
  - Create C matrix – O(mk)
  - Create R matrix – O(nk)
- Construct U
  - Compute C^TC – O(mk^2)
  - SVD of C^TC – O(k^3)
  - Compute A and B – O(k^3)
  - U = AB^T – O(k^3)

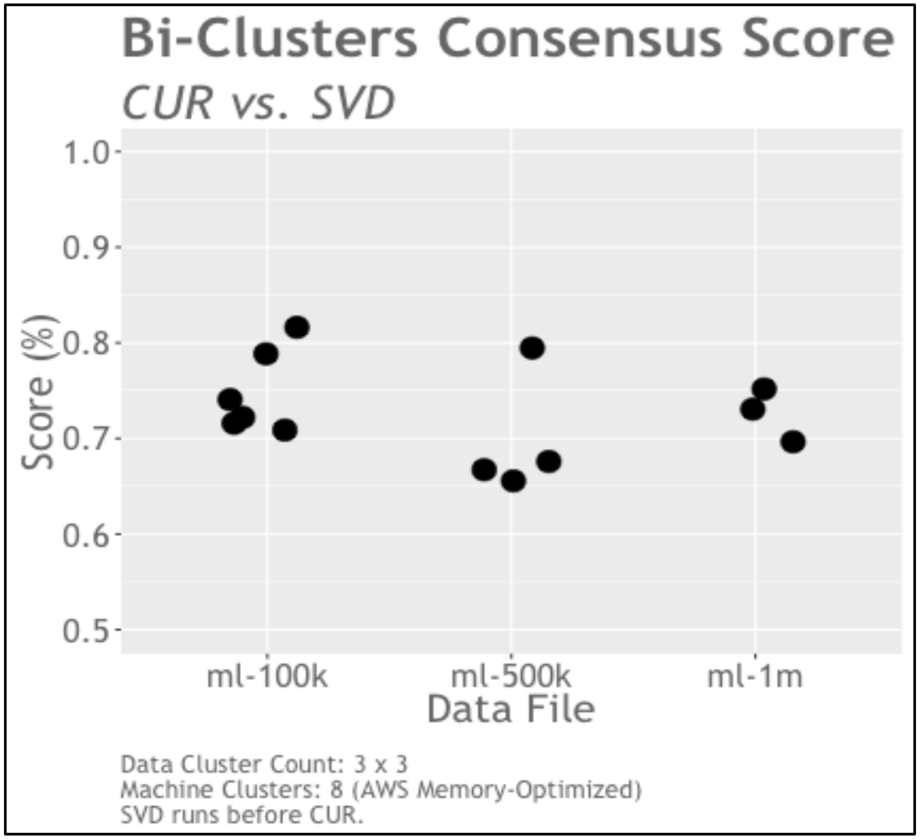
Distributed (communication cost and computation time)

- Build C and R:
- Generate probabilities – O(mn + p) cost, O(max dense) time
  - **Create 2 RDDs by Row/Col partition – O(mn) cost, AtoA**
  - **Both instances: reduce to Row/Col sums — O(max dense) time, no communication**
  - One instance: reduce Row sum to total – O(p) cost, O(log p) time
  - Broadcast total to calculate probs – O(p) cost, O(log p) time
- Create C / R matrices
  - Locally sample k rows/cols – O(k)
  - Broadcast sample to RDDs – O(pk) cost, O(k log p) time
- Construct U
  - Same as Serial (less opportunity to distribute)

Biclustering: Distributed CUR vs SVD - Empirics



Data Cluster Count: 3 x 3
Machine Clusters: 8 (AWS Memory-Optimized)
SVD runs before CUR.



Data Cluster Count: 3 x 3
Machine Clusters: 8 (AWS Memory-Optimized)
SVD runs before CUR.