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

Biclustering also has potential in other areas and has added interpretability

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

CUR – less accurate but light,
More interpretable*

* As archetypal users and movies

Movie Ratings
sparse / huge

Source: Source Code for Biology Medicine (April 2013) – "The non-negative matrix factorization toolbox for biological data mining"
Review of SVD: $A = U \Sigma V^T$

- **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)
Background on A = CUR

- **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**

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

A sparse / huge

C sparse big

R sparse / big

U dense / small

Pinv( )
Design Decisions for Distributed CUR

Key Design Decision:
Distribute two instances of A avoiding future all-to-all communications

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.
### 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, \( O(\text{max dense}) \) time
    - *Create 2 RDDs by Row/Col partition* – \( O(mn) \) cost, AtoA
    - *Both instances: reduce to Row/Col sums* — \( O(\text{max dense}) \) 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

**Distributed Run Times**

*SVD and CUR Decompositions*

- **Run Time (seconds)**
  - ml-100k
  - ml-500k
  - ml-1m

- **Group**
  - CUR
  - SVD

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

**Bi-Clusters Consensus Score**

*CUR vs. SVD*

- **Score (%)**
  - ml-100k
  - ml-500k
  - ml-1m

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