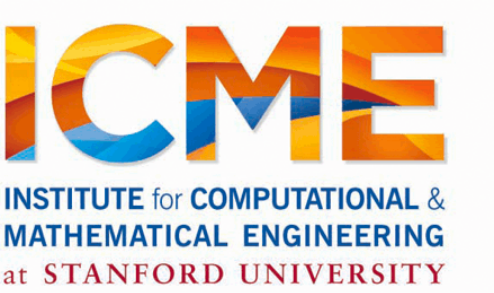
Matrix Factorization

Reza Zadeh



@Reza_Zadeh | http://reza-zadeh.com

Outline

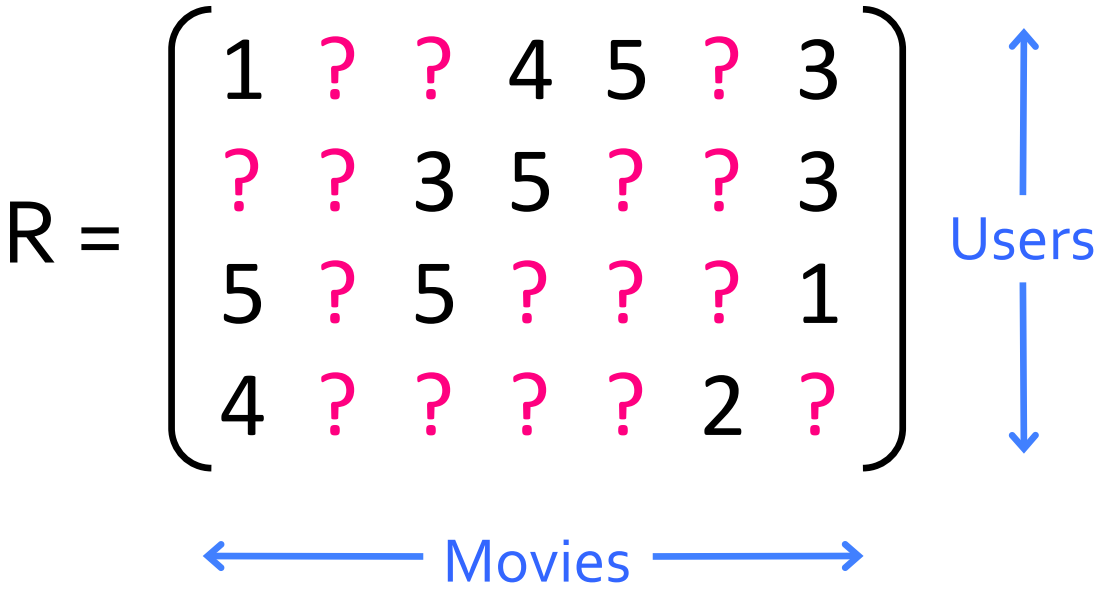
Matrix Factorization (collaborative filtering)

Sparse subspace embedding

Stochastic Gradient Descent (on the board)

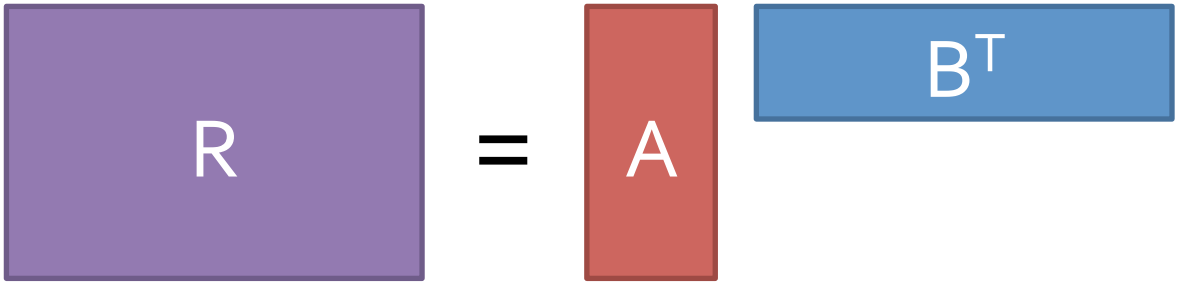
Collaborative Filtering

Goal: predict users' movie ratings based on past ratings of other movies



Model and Algorithm

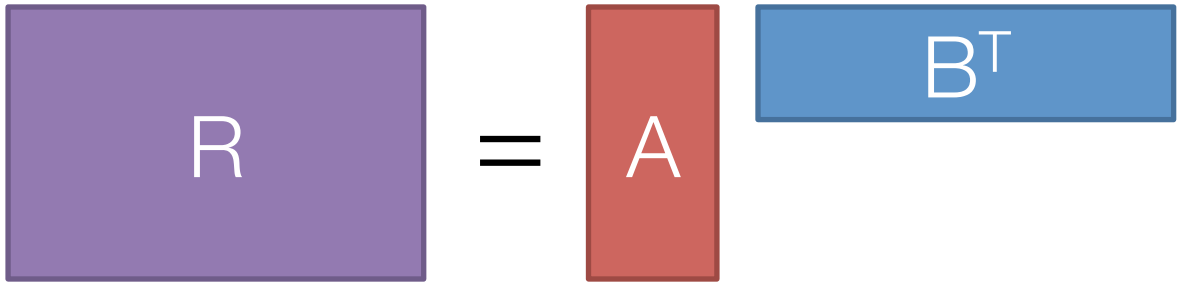
Model R as product of user and movie feature matrices A and B of size U×K and M×K



Alternating Least Squares (ALS)

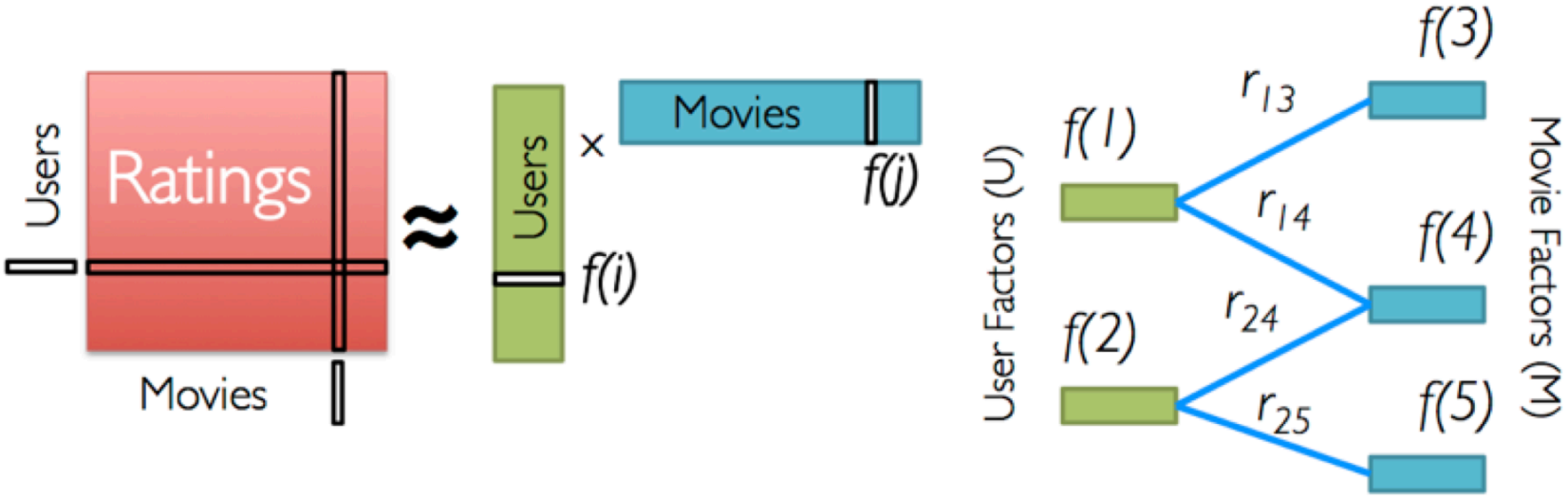
- » Start with random A & B
- » Optimize user vectors (A) based on movies
- » Optimize movie vectors (B) based on users
- » Repeat until converged

Alternating Least Squares



1. Start with random A_1, B_1
2. Solve for A_2 to minimize $||R - A_2 B_1^T||$
3. Solve for B_2 to minimize $||R - A_2 B_2^T||$
4. Repeat until convergence

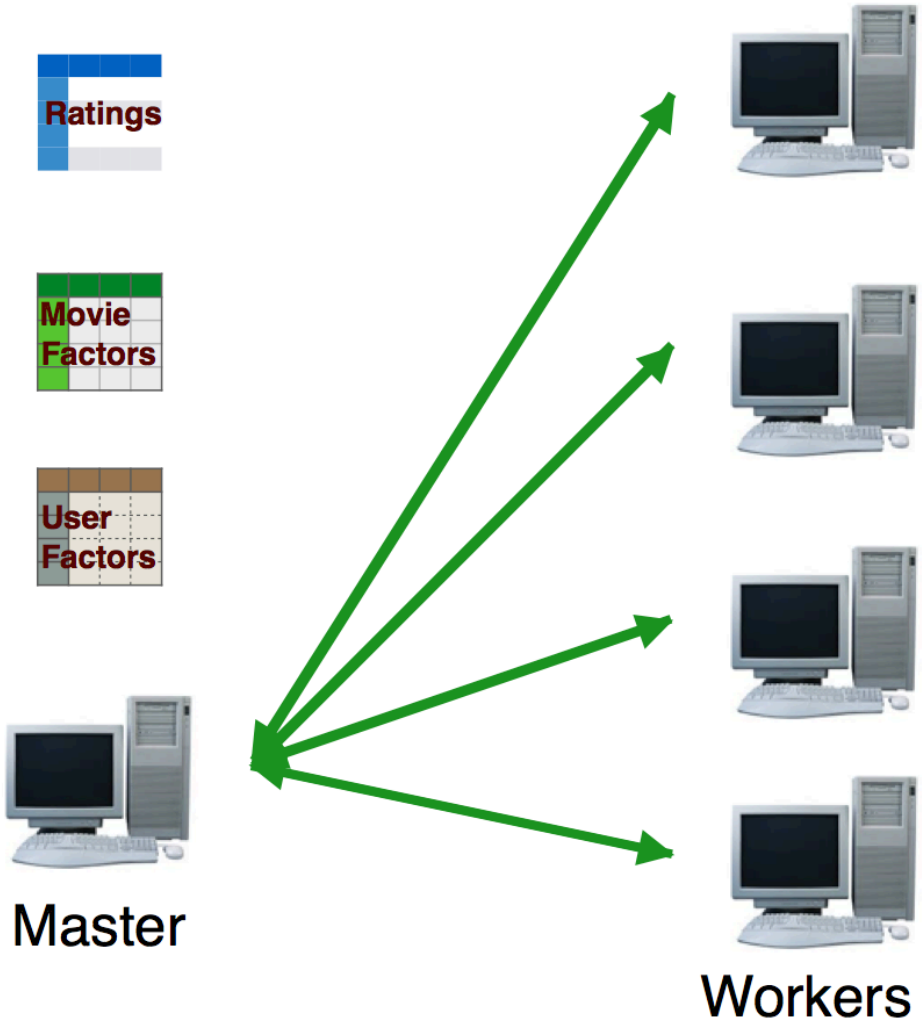
Optimization problem



Iterate:

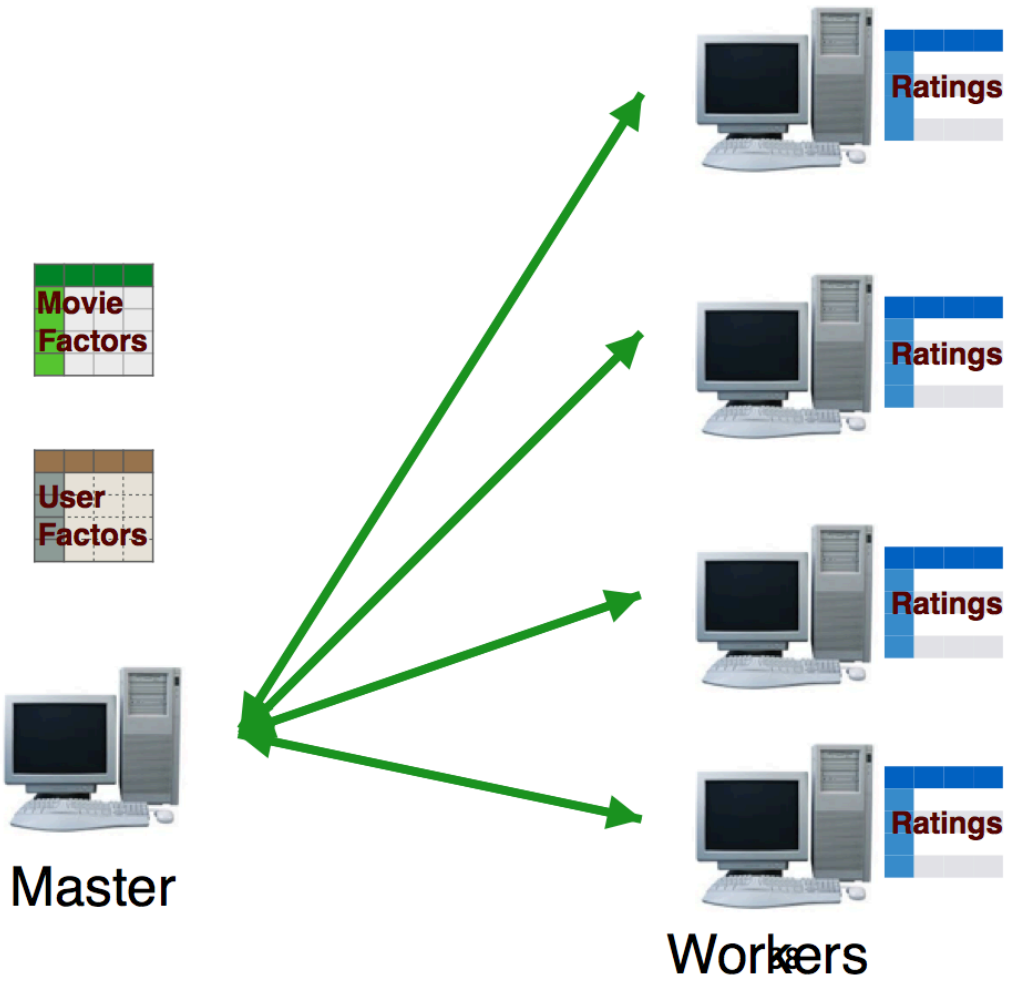
$$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathbf{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$$

Attempt 1: Broadcast All



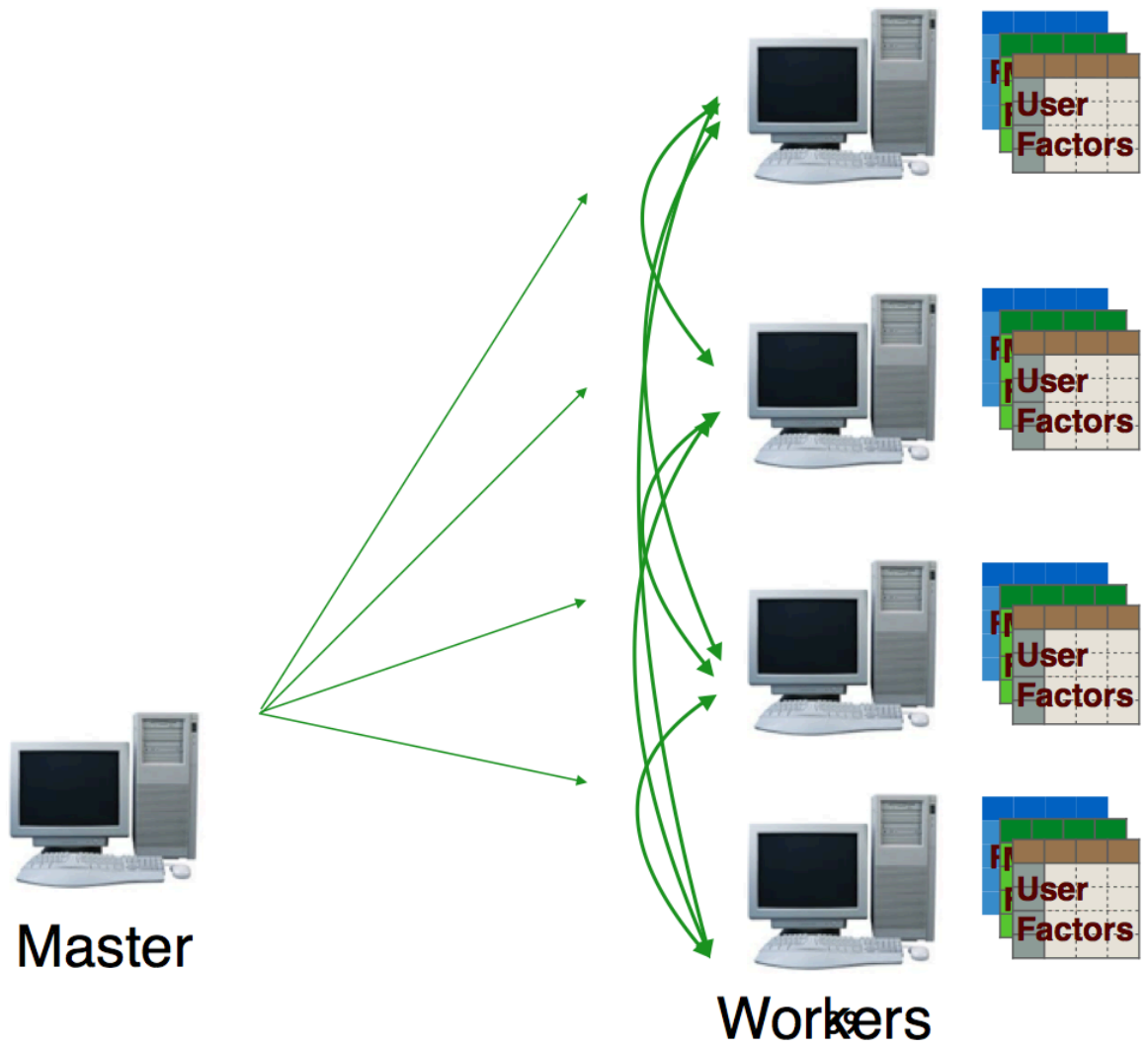
- Master loads (small) data file and initializes models.
- Master broadcasts data and initial models.
- At each iteration, updated models are broadcast again.
- Works OK for small data.
- Lots of communication overhead - doesn't scale well.

Attempt 2: Data Parallel



- Workers load data
- Master broadcasts initial models
- At each iteration, updated models are broadcast again
- Much better scaling
- Works on large datasets
- Works well for smaller models. (low K)

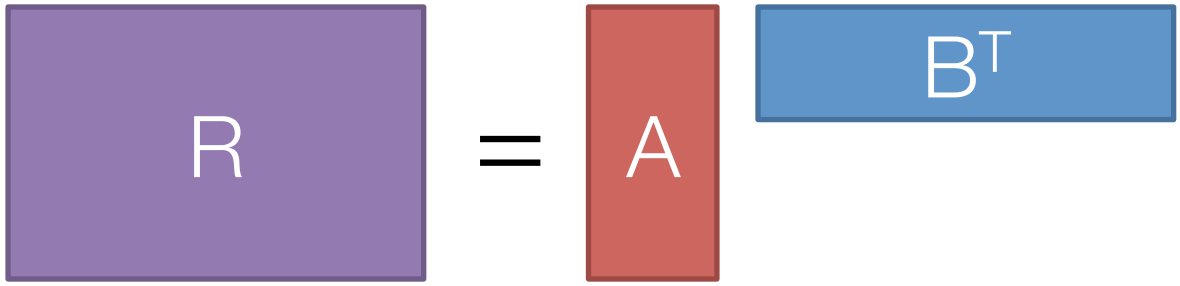
Attempt 3: Fully Parallel



- Workers load data
- Models are instantiated at workers.
- At each iteration, models are shared via join between workers.
- Much better scalability.
- Works on large datasets

ALS on Spark

Matei Zaharia,
Joey Gonzales,
Virginia Smith

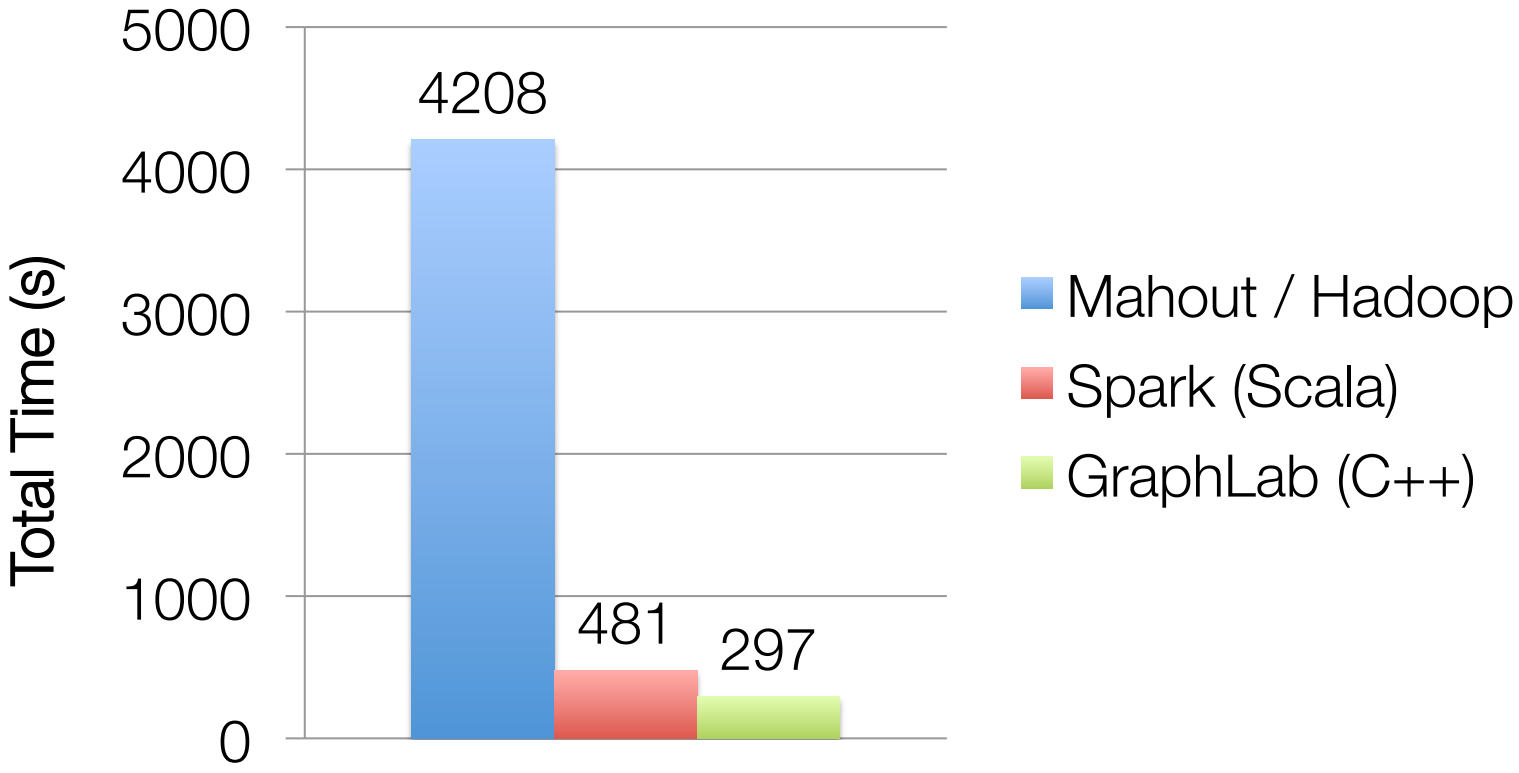


Cache 2 copies of R in memory, one partitioned by rows and one by columns

Keep A & B partitioned in corresponding way

Operate on blocks to lower communication

ALS Results


5000
4000
3000
2000
1000
0
Total Time (s)
4208
481
297
Mahout / Hadoop
Spark (Scala)
GraphLab (C++)

Sparse subspace embedding

Sparse Subspace Embedding

(Φ has one nonzero per column)

```
A = # RDD of vectors, one per row
s = 10000   # embedding dimension
S = A.map(lambda row: (randint(1, s), gauss(0, 1) * row)) \
     .reduceByKey(lambda a, b: a + b) \
     .values()
```

[Clarkson and Woodruff, STOC '13]

Stochastic Gradient Descent (on the board, in the notes)