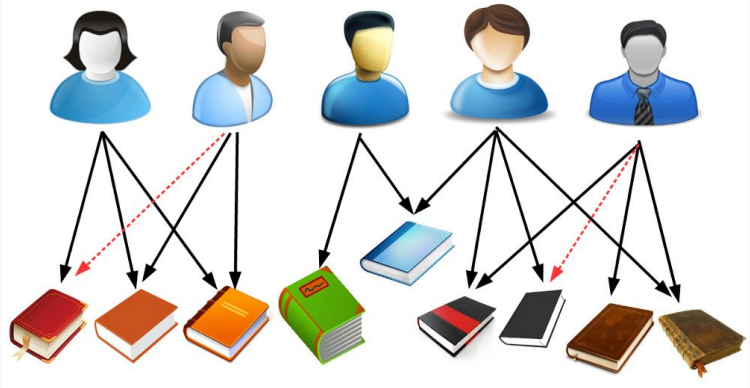
Low-rank Matrix Factorization

With Distributed Stochastic Gradient Descent

Introduction

- Important problem for designing recommender systems and analyze big data
- Previous algorithms: SSVD, ALS
- Spark MLlib library uses distributed ALS



Matrix Factorization

Given an m x n matrix **V**, want to find

an m x r matrix **W** and an r x n matrix **H** such that

$$\mathbf{V} \cong \mathbf{WH}$$

i.e. **W** and **H** that minimize a loss function

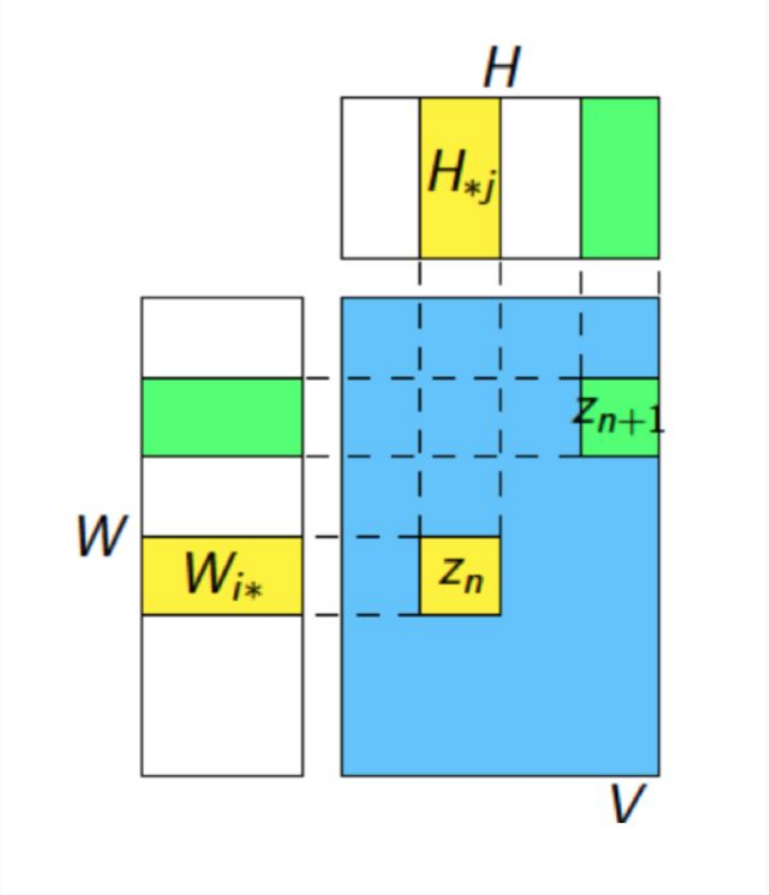
$$L(\mathbf{V}, \mathbf{W}, \mathbf{H})$$

SGD, SSGD, and DSGD

- Stochastic gradient descent
- Stratified stochastic gradient descent

  $L(\theta) = w_1 L_1(\theta) + w_2 L_2(\theta) + \ldots + w_q L_q(\theta)$

- Distributed stochastic gradient descent



Stratum Selection

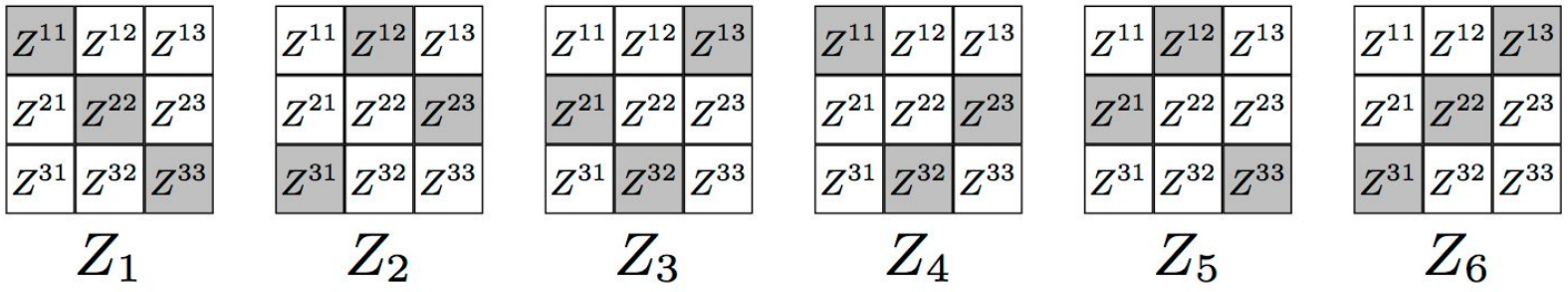


Figure 1: Strata for a 3×3 blocking of training matrix $\boldsymbol{Z}$

Implementation

- Choose a stratum and assign corresponding blocks of **V**, **W**, and **H** to the same machine ID.
  - Ensure this by using Spark transformations such as cogroup(), partitionBy(), and mapPartitions().
- Each machine performs sequential SGD on its block of **V**, **W**, and **H**.
- Blocks are interchangeable so we can just concatenate results instead of doing any aggregation

Results

Dataset: Movielens 1 M ratings, 6000 users, 4000 movies

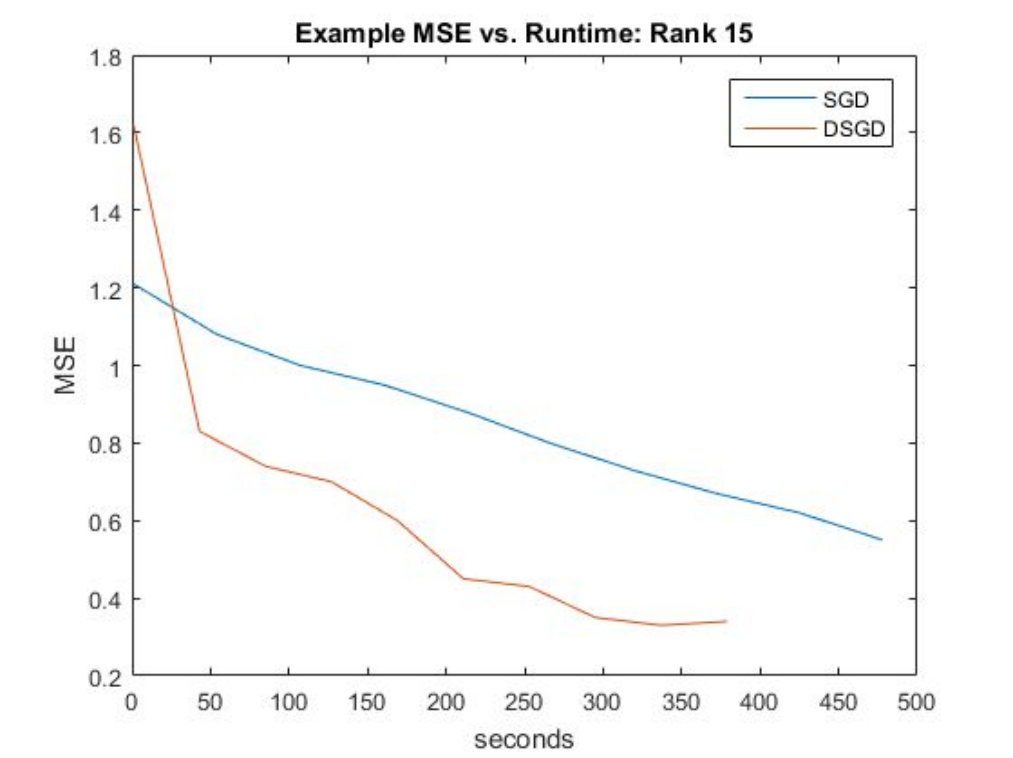
Factor Size Runtime Dependence (runtime until convergence in factors)

Latent Factor Rank	Spark ALS	DSGD	Sequential SGD
5	27.5 s	352.2 s	530.2 s
10	30.1 s	428.2 s	> 600 s
50	271.3 s	464.2 s	> 600 s

*Sequential SGD can outperform the DSGD when the matrix is very sparse, but this dataset is extremely dense
*DSGD and ALS were run on databricks cluster

Results

Sequential SGD vs. DSGD (MSE vs. Runtime)



References

1. Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the Netflix Prize. In AAIM, pages 337–348, 2008.
2. R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. KDD, 2011.

Grouping vs Broadcasting

Grouping

- Have to communicate the matrices at every iteration, but in smaller size.

Broadcasting

- Broadcast **V** only at the beginning, may perform better given enough space.
- **W** and **H** still need to be updated globally.