Implement Distributed Alternating Least Squares Algorithm for Matrix Completion

Varun Gandhi (vg292)
Netflix Problem

- $V$: $m \times n$ matrix
- complete the matrix

- $W$: $m \times r$ (row-factor matrix)
- $H$: $r \times n$ (column-factor matrix)
- $W \times H$ approx $V$
- Loss function $(V_{ij} - WH_{ij})^2$
Motivation

Large applications involve matrices with

- millions of rows x columns;
- billions of entries

To achieve high-performance

- parallel & distributed factorisation
- keep the loss to minimum
Algorithm

Sequential Computation

• Initial point $W_0$ and $H_0$
• ALS solved for every row & column

\[
\text{Compute } W_{n+1}: (\forall i) \quad W_{i*} H_n^{(i)} = V_{i*},
\]
\[
\text{Compute } H_{n+1}: (\forall j) \quad W_{n+1}^{(j)} H_{*j} = V_{*j},
\]

Parallel Computation

• Parallelise computation for rows and columns respectively
Distributed Computation

- Partition (block) the matrix with $m_b \times n_b$ matrices
- every node updates a matrix block

Why Spark?

- In-memory algorithm
- Matrix versions cached in memory
Progress

• Revising all linear algebra concepts
• Getting familiar with Scala and Spark
• Trying examples in Python