UNIVERSITY OF CAMBRIDGE

Implement Distributed Alternating Least Squares Algorithm for Matrix Completion

Varun Gandhi (vg292)

Computer Laboratory

Netflix Problem

- V: m*n matrix
- complete the matrix

$$\begin{array}{c} \\ \textit{Alice} \\ \textit{Bob} \\ \textit{Charlie} \end{array} \begin{array}{c} \begin{array}{ccc} \textit{Avatar} & \textit{The Matrix} & \textit{Up} \end{array} \\ \begin{pmatrix} ? & 4 & 2 \\ 3 & 2 & ? \\ 5 & ? & 3 \end{pmatrix} \end{array}.$$

- W: m*r (row-factor matrix)
- H: r*n (column-factor matrix)
- W*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

$$\text{Compute } \boldsymbol{W}_{n+1}\text{: } (\forall i)\ \underline{\boldsymbol{W}_{i*}}\boldsymbol{H}_n^{(i)} = \boldsymbol{V}_{i*},$$

$$\text{Compute } \boldsymbol{H}_{n+1}\text{: } (\forall j)\ \boldsymbol{W}_{n+1}^{(j)}\underline{\boldsymbol{H}_{*j}} = \boldsymbol{V}_{*j},$$

Parallel Computation

- Parallelise computation for rows and columns respectively

Algorithm

Distributed Computation

- Partition (block) the matrix with $m_b * n_b$ 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