Discussion 2

Alejandro D. Osborne

June 23, 2019

Thoughts

The standard approach to matrix factorization based collaborative filtering treats the entries in the user-item matrix as explicit preferences given by the user to the item. It is common in many real-world use cases to only have access to implicit feedback (e.g. views, clicks, purchases, likes, shares etc.). The approach used in MLlib to deal with such data is taken from Collaborative Filtering for Implicit Feedback Datasets. Essentially instead of trying to model the matrix of ratings directly, this approach treats the data as a combination of binary preferences and confidence values.

Spark is a cluster-computing framework, which means that it competes more with MapReduce than with the entire Hadoop ecosystem. For example, Spark doesn't have its own distributed filesystem, but can use HDFS. Spark uses memory and can use disk for processing, whereas MapReduce is strictly disk-based. Spark is simpler and usually much faster than Mapreduce for the usual Machine learning and Data Analytics applications. The speaker Christopher Johnson says that Spark is the clear choice for running the recommender programs and provides a comparison of time it takes by Hadoop MapReduce and Spark, Spark completes the execution 7 times faster.

In Full gridify scheme, a lot of intermediate data is sent over the wire for each iteration and a large I/O overhead compared to the half gridify scheme, but has a potential for requiring less local memory comapred to half gridify. For a half gridify scheme, ratings get cached and never shuffled. Once item vectors are joined with ratings partitions each partition has enough information to solve optimal user and vectors without any additional shuffling.

Key Lessons

- Running with larger datasets often results in failed executors and the job never fully recovers. Ideal size is 20% of the data set
- Kryo serialization is faster than Java serialization with some caveats like writing custom serializers