Matrix Factorization with SGD using Spark and Redis

Linquan Chen, Yuankun Chang, Vinodh Paramesh

1. Problem and Application

1.1 Problem

The problem we are trying to solve is that of Matrix Factorization . Formally, Given a **M* N** matrix "**V**" with the rank "**r**", find "**M*r**" matrix "**W**" and "**r * N**" matrix "**H**" , such that "**V= W H**" . Of all the techniques to solve matrix factorization problem, Gradient Descent based algorithms are most suitable. However, naive gradient descent based algorithms are not suitable for large datasets where the matrix size is huge.

Stochastic Gradient Descent(SGD) is a special kind of "Gradient descent algorithm" in which the true gradient is approximated by a gradient at a single point rather than whole of dataset. This makes stochastic gradient descent apt for the matrices with large number of rows and columns. However, Stochastic gradient descent is an iterative algorithm and is inherently sequential in nature. Though SGD is not embrassingly parallel, we can exploit the special structure of the factorization problem to obtain a version of SGD that can be run in parallel setting and be scaled to very large matrices with better performance. Our goal is to utilize the parallel frameworks such as "Spark" and in memory databases such as redis to solve this factorization problem with high efficiency and in a parallel way.

1.2 Application

Low rank matrix factorization forms a core technique in recommendation systems. For example, in movie recommendation systems, matrix 'W' could be a matrix of user features and 'H' could be movie features and final matrix "V" is the matrix of movie ratings by users.

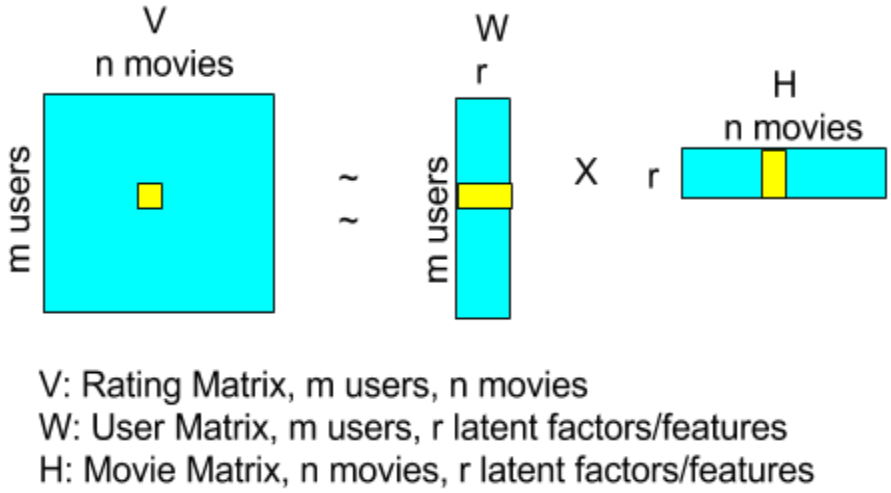

Figure 1. Matrix factorization in movie recommendation system.

As shown at Figure 1, we are trying to fill up the matrix "V" by factorizing it into W and H. That filled up entity rating can be used to recommend the movies to the user. As a matter of fact , we would be using the dataset from 'movielens' to validate the optimization we are attempting to bring on.

2. Platform and Techniques

2.1 Platform: Cloud Computing platform.

Cloud computing platform is the most famous platform to do parallel computing. And it is far more faster than multicore and manycore platform. So we want to apply cloud computing platform in our project.

2.2 Techniques

1) Algorithm

We want to use the stochastic gradient descent(SGD) algorithms for matrix factorization.

2) Spark

Spark is an open source cluster computing framework. It uses in-memory primitives that allow it to perform faster than traditional MapReduce for certain applications. It allows for a distributed dataset to remain in-memory in the nodes of a cluster during various stages of computation. Resilient distributed Datasets(RDD) represents a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. It is much faster than HDFS and MapReduce since data is processed in memory.

3) Redis

Redis is an open source (BSD licensed), in-memory data structure store, used as database, cache and message broker. It supports data structures such as strings, hashes, lists, sets,sorted sets with range queries, bitmaps and geospatial indexes with radius queries. Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster. There is a spark redis connector called spark-redis. We can try to use it and make the optimize delay.

3. Expectations and Improvements in throughput

We will use a few different sizes of the MovieLens datasets, each has millions of movies ratings made by thousands(to hundreds of thousands) of anonymous users of thousands(to tens of thousands) of movies. You can access the datasets at http://grouplens.org/datasets/movielens.

Table 1 shows the sequential performance implemented using Python. Our main expectations is to improve the runtime and runtime-memory product. For the runtime aspect, we plan to speed up 2x or more with limited resources. Because when using the Spark Cluster, we can achieve better performance with more workers. However, this method will waste lots of computing resources. Therefore, in order to improve the resource utilization, we restrict another criteria, runtime-memory product, which is the product of runtime(in seconds) and the provisioned memory. Our goal is to make sure that the runtime-memory is less than 1.5 times than the sequential one.

Table 1 Sequential Performance of Matrix Factorization with SGD

Dataset	**Ratings**	**Users**	**Movies**	**Rank**	**Provisioned memory**	**Runtime (seconds)**	**Runtime -memory product**	**RMSE**
ML_1M	1000000	6000	4000	30	7.5G	360	2250	0.896
ML_10M	10000000	72000	10000	100	7.5G	4800	36000	0.864
ML_22M	22000000	240000	33000	500	7.5G	36000	270000	0.8586

4. Project Timeline

DATE	MILESTONE
By 25th March	Pre-process and get data ready, perform literature survey and read up the techniques and current state, get hands on technologies
By March 29th	baselining the current runtime and performance with existing technique
By April 6th	Apply optimization and get new code base ready
By April 12th	Compare performances , review with faculty, mid way report
By April 16th	Refine optimizations and prepare for working demo for realistic use case
By april 20th	Final report submission with code base

5. Infrastructure Requirements

a. Amazon Web Service(Spark)
b. Spark-redis connector