

**Scalable Distributed Nonnegative Matrix Factorization with Block-Wise Updates**

**Abstract:**

Nonnegative Matrix Factorization (NMF) has been applied with great success on a wide range of applications. As NMF is increasingly applied to massive datasets such as web-scale dyadic data, it is desirable to leverage a cluster of machines to store those datasets and to speed up the factorization process. However, it is challenging to efficiently implement NMF in a distributed environment. In this paper, we show that by leveraging a new form of update functions, we can perform local aggregation and fully explore parallelism. Therefore, the new form is much more efficient than the traditional form in distributed implementations. Moreover, under the new form of update functions, we can perform frequent updates and lazy updates, which aim to use the most recently updated data whenever possible and avoid unnecessary computations. As a result, frequent updates and lazy updates are more efficient than their traditional concurrent counterparts. Through a series of experiments on a local cluster as well as the Amazon EC2 cloud, we demonstrate that our implementations with frequent updates or lazy updates are up to two orders of magnitude faster than the existing implementation with the traditional form of update functions.

**Existing System:**

Many practitioners nowadays have to deal with NMF on massive datasets. For example, recommendation systems in web services such as Netflix have been dealing withNMF on web-scale dyadic datasets, which involve millions of users, millions of movies, and billions of ratings. For such web-scale matrices, it is desirable to leverage a cluster of machines to speed up the factorization process. MapReduce and its variants has emerged as a popular distributed framework for data intensive computation.

It provides a simple programming model where a user can focus on the computation logic without worrying about the complexity of parallel computation. Prior approaches of handling NMF onMapReduce usually pick an

NMF algorithm and then focus on implementing matrix operations on Map Reduce.

**Proposed System:**

We present a new form of factor matrix update functions. This new form operates on blocks of matrices. In order to support the new form, we partition the factor matrices into blocks along the short dimension to maximize the parallelism and split the original matrix into corresponding blocks.

The new form allows us to update distinct blocks independently and simultaneously when updating a factor matrix. It also facilitates a distributed implementation.

Different blocks of one factor matrix can be updated in parallel, and can be distributed in memories of all machines of a cluster and thus avoid overflowing the memory of one single machine. Storing factor matrices in memory can support random access and local aggregation. As a result, the new form of update functions leads to an efficient MapReduce implementation. We illustrate that the new form works for NMFs with a wide class of loss functions.