BIGtensor (Hadoop)
User Guide

v 1.0.1

Data Mining Lab
Seoul National University

Contents

1. Overview

1.1. General Information

BIGtensor (Hadoop) is a large scale tensor mining package running on the Hadoop (MapReduce) platform. BIGtensor (Hadoop) supports various tensor operations, tensor generations, and tensor factorizations.

Version: 1.0.1

Date: Feb 19th, 2020

1.2. Document Version

Version	Content	Author
1.0.0	First draft of guide	ByungSoo Jeon
1.0.1	Revision of draft	Dawon Ahn

2. Installation

2.1. Environment

To run BIGtensor (Hadoop), Hadoop and Java should be installed in your system in advance.

- Hadoop version 1.0.4 or greater from http://hadoop.apache.org/
- Java version 1.6.x or greater, preferably from Sun

BIGtensor (Hadoop) can run in any machine that supports Java and Hadoop, but the shell scripts and code packaging scripts work easily in Linux or Unix machines.

2.2. Download

Download ‘BIGTensor-hadoop.tar.gz’ from http://datalab.snu.ac.kr/bigtensor/

Extract the file, then the directory ‘BIGtensor-hadoop’ will be created.

Cd to the BIGtensor-hadoop directory, then you are done.

2.3. Install

Before you use it, please check that the script files are executable. If not, you may manually modify the permission of scripts (chmod +x *.sh) or you may type "make install".

3. Operations

3.1. Tensor Generation

BIGtensor (Hadoop) provides 4 types of tensor generations.

Generation	Description
Ones	Tensor generation where all tensor values are set to 1.
Random	Tensor generation where tensor values are set randomly.
FromParafacFactors	Tensor generation from factor matrices and lambda vector (Parafac).
FromTuckerFactors	Tensor generation from factor matrices and core tensor (Tucker).

3.2. Tensor Factorization

BIGtensor (Hadoop) supports 5 types of general tensor factorizations: PARAFAC, Nonnegative PARAFAC, Tucker, Nonnegative Tucker, and coupled matrix-tensor factorization (PARAFAC).

Factorization	Description
PARAFAC	PARAFAC
PARAFAC-NN	Nonnegative PARAFAC
Tucker	Tucker
Tucker-NN	Nonnegative Tucker
Coupled matrix-tensor factorization	Coupled matrix-tensor factorization (PARAFAC)

3.3. Tensor Operation

BIGtensor (Hadoop) supports various operations for tensors. They are divided into tensor-tensor operations (operands are two tensors) and tensor operations (operand is a tensor). Tensor operations are described in the following sections.

3.3.1.Tensor-Tensor Operation

Tensor-Tensor Operation	Description
BinaryOperations	Binary operations (and, or, xor) between two binary tensors.
FundamentalArithmetics	Four Fundamental rules of arithmetics (+, -, *, /) operations between two tensors.
EqualTo	Comparison operation whether two tensors are identical or not.
NModeProduct	n-mode product between two tensors.

3.3.2.Tensor Operation

Tensor Operation	Description
Collapse	Collapsing a mode of a tensor.
ConvertSign	Converting the sign of all values in a tensor.
ConvertToBinaryTensor	Converting a tensor into binary tensor.
Find	Finding elements satisfying certain conditions or index in a tensor.
Matricization	Matricization of a tensor to certain mode.
Norm	Computing L1-norm or L-F norm of a tensor.
PermutationOfModes	Permuting the order of modes of a tensor (transpose in a matrix).
ScalarOperation	Scalar operations (+, -, *, /) for a tensor and a scalar value.
Scaling	Scaling up a mode of a tensor.

4. Running

4.1. Preparing a tensor file

Prepare a tensor file (e.g. test_tensor) in a HDFS directory;

‘hadoop fs –put test_tensor tensor_path/test_tensor’.

Tensors are stored in the sparse tensor format. The left columns have indices of each mode, and the last column has the value of specified position (i_1, i_2, …, i_N, value). Each column is separated by a tab and **the indices of tensor always begin with the number 1, not 0.**

For example, the following is the sample tensor file ‘src/test/resources/testInput/test_tensor’

```
1   1   1   0.4
4   3   1   -0.125
2   3   1   0.36
1   2   1   0.64
5   1   2   -0.23
4   4   2   0.843
4   2   3   0.74
2   1   3   0.24
5   3   3   0.433
1   4   3   -0.5
```

4.2. Running scripts

You can run each method with corresponding shell scripts. Here is the list of the methods, shell scripts, and corresponding demo scripts.

Methods	Running Script	Demo Script
Tensor Generation		
Ones	run_gen_ones.sh	do_gen_ones.sh
Random	run_gen_rand.sh	do_gen_rand.sh
FromParafacFactors	run_gen_fpf.sh	do_gen_fpf.sh
FromTuckerFactors	run_gen_ftf.sh	do_gen_ftf.sh
Tensor Factorization		
PARAFAC	run_parafac.sh	do_parafac.sh
PARAFAC-NN	run_parafacnn.sh	do_parafacnn.sh
Tucker	run_tucker.sh	do_tucker.sh
Tucker-NN	run_tuckernn.sh	do_tuckernn.sh
Coupled matrix-tensor factorization	run_cmtf.sh	do_cmtf.sh
Tensor-Tensor Operation		
BinaryOperations	run_binary_op.sh	do_binary_op.sh
FundamentalArithmetics	run_ffra_op.sh	do_ffra_op.sh
EqualTo	run_equal_to.sh	do_equal_to.sh
NModeProduct	run_nmode_product.sh	do_nmode_product.sh
Tensor Operation		
Collapse	run_collapse.sh	do_collapse.sh
ConvertSign	run_conv_sign.sh	do_conv_sign.sh
ConvertToBinaryTensor	run_conv_bin.sh	do_conv_bin.sh
Find	run_find_cond.sh run_find_elem.sh	do_find_cond.sh do_find_elem.sh
Matricization	run_matricization.sh	do_matricization.sh
Norm	run_norm.sh	do_norm.sh
PermutationOfModes	run_permute_mode.sh	do_permute_mode.sh
ScalarOperation	run_scalar_op.sh	do_scalar_op.sh
Scaling	run_scaling.sh	do_scaling.sh

The ‘running script’ is the script you need to use to run each method. It requires several parameters which will be described next. The ‘demo script’ is the script to tell you how to use the ‘running script’. The demo scripts do not require any parameters. Just type the demo script name and it will run. Or, if you simply type make in the installation directory, the demo of running Parafac factorization method on the example tensor ‘test.tensor’ will be executed.

All running scripts with no input path parameter do not require any input files. The output is stored in the specified output path on HDFS. The result tensors and matrices are stored in the sparse tensor format. The left columns have indices of each mode, and the last column has the value of specified position (i_1, i_2, …, i_N, value). Each column is separated by a tab and the indices of tensor begin with the number 1, not 0. The example is the same as in section 4.1.

4.2.1.Tensor Generation

4.2.1.1. run_gen_ones.sh

You don’t need any preparation to execute ‘run_gen_ones.sh’.

./run_gen_ones.sh [output path] [dim_1:..:dim_N (tensor)] [# of reducers]

- output path: output path of the result tensor
- dim_1:..:dim_N (tensor): size of the result tensor
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.1.2. run_gen_rand.sh

You don’t need any preparation to execute ‘run_gen_rand.sh’.

./run_gen_rand.sh [output path] [dim_1:..:dim_N (tensor)] [# of reducers]

- output path: output path of the result tensor
- dim_1:..:dim_N (tensor): size of the result tensor
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.1.3. run_gen_fpf.sh

You need to upload to the HDFS directory factor matrices stored in the sparse tensor format.

./run_gen_fpf.sh [order] [rank] [dim_1:..:dim_N (tensor)] [factor path] [output path] [# of reducers]

- order: number of modes
- rank: number of ranks
- dim_1:..:dim_N (tensor): size of the result tensor
- factor path: input path of the factor matrices
- output path: output path of the result tensor
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.1.4. run_gen_ftf.sh

You need to upload to the HDFS directory factor matrices stored in the sparse tensor format.

./run_gen_ftf.sh [order] [dim_1:..:dim_N (core tensor)] [dim_1:..:dim_N (tensor)] [factor path] [output path] [# of reducers]

- order: number of modes
- dim_1:..:dim_N (core tensor): size of the core tensor
- dim_1:..:dim_N (tensor): size of the result tensor
- factor path: input path of the factor matrices
- output path: output path of the result tensor
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.2.Tensor Factorization

4.2.2.1. run_parafac.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_parafac.sh [dim_1:..:dim_N (tensor)] [rank] [# of reducers] [max iteration] [tensor path] [output path]

- dim_1:..:dim_N (tensor): size of the input tensor
- rank: number of ranks
- # of reducers: number of reducers to use
- max iteration: maximum number of iterations
- tensor path: path of the input tensor
- output path: output path of the result factor matrices

The output file is in the output path on HDFS.

4.2.2.2. run_parafacnn.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_parafacnn.sh [dim_1:..:dim_N (tensor)] [rank] [# of reducers] [max iteration] [tensor path] [output path]

- dim_1:..:dim_N (tensor): size of the input tensor
- rank: number of ranks
- # of reducers: number of reducers to use
- max iteration: maximum number of iterations
- tensor path: path of the input tensor
- output path: output path of the result factor matrices

The output file is in the output path on HDFS.

4.2.2.3. run_tucker.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_tucker.sh [dim_1:..:dim_N (tensor)] [dim_1:...:dim_N (core tensor)] [# of reducers] [max iteration] [tensor path] [output path]

- dim_1:..:dim_N (tensor): size of the input tensor
- dim_1:...:dim_N (core tensor): size of the core tensor
- # of reducers: number of reducers to use
- max iteration: maximum number of iterations
- tensor path: path of the input tensor
- output path: output path of the result factor matrices

The output file is in the output path on HDFS.

4.2.2.4. run_tuckernn.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_tuckernn.sh [dim_1:..:dim_N (tensor)] [dim_1:...:dim_N (core tensor)] [# of reducers] [max iteration] [tensor path] [output path]

- dim_1:..:dim_N (tensor): size of the input tensor
- dim_1:...:dim_N (core tensor): size of the core tensor
- # of reducers: number of reducers to use

- max iteration: maximum number of iterations
- tensor path: path of the input tensor
- output path: output path of the result factor matrices

The output file is in the output path on HDFS.

4.2.2.5. run_cmtf.sh

You need to upload to the HDFS directory coupled tensor-matrix stored in the sparse tensor format.

./run_cmtf.sh [dim_1:dim_2:...:dim_N (dimensions of tensor)] [mode_1:mode_2:...:mode_N (coupled modes)] [dim_1:dim_2:...:dim_N (dimensions of matrix)] [Rank] [number of reducer] [max iteration] [tensor_path] [path_1:path2:...:path_N (paths of coupled matrices)] [output_path]

- dim_1:dim_2:...:dim_N (dimensions of tensor): size of the input tensor
- mode_1:mode_2:...:mode_N (coupled modes): coupled modes with the input tensor
- dim_1:dim_2:...:dim_N (dimensions of matrix): size of the input matrix
- rank: number of ranks
- number of reducer: number of reducers to use
- max iteration: maximum number of iterations
- tensor_path: path of input tensor
- path_1:path2:...:path_N (paths of coupled matrices): paths of coupled input matrices
- output_path: output path of the result factor matrices

The output file is in the output path on HDFS.

4.2.3.Tensor-Tensor Operation

4.2.3.1. run_binary_op.sh

You need to upload to the HDFS directory two tensors stored in the sparse tensor format.

./run_binary_op.sh [input path] [input path2] [output path] [operation type] [# of reducers]

- input path: input path of the first tensor
- input path2: input path of the second tensor
- output path: output path of the operation result
- operation type: type of the binary operation (e.g. and ‘&’, or ‘|’, xor ‘^’)
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.3.1. run_ffra_op.sh

You need to upload to the HDFS directory two tensors stored in the sparse tensor format.

./run_ffra_op.sh [input path] [input path2] [output path] [operation type] [# of reducers]

- input path: input path of the first tensor
- input path2: input path of the second tensor
- output path: output path of the operation result
- operation type: type of the fundamental arithmetic operation (e.g. ‘+’. ‘-‘, ‘*’, ‘/’)
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.3.2. run_equal_to.sh

You need to upload to the HDFS directory two tensors stored in the sparse tensor format.

./run_equal_to.sh [input path] [input path2] [# of reducers]

- input path: input path of the first tensor
- input path2: input path of the second tensor
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.3.3. run_nmode_product.sh

You need to upload to the HDFS directory two tensors stored in the sparse tensor format.

./run_nmode_product.sh [# of dimensions1] [dim_1:dim_2:...:dim_N (dimensions of tensor1)] [mode_1:mode_2:...:mode_N (product modes1)] [# of dimensions2] [dim_1:dim_2:...:dim_N (dimensions of tensor2)] [mode_1:mode_2:...:mode_N (product modes2)] [input path] [output path] [# of reducers]

- # of dimensions1: number of the dimensions of the first tensor
- dim_1:dim_2:...:dim_N (dimensions of tensor1): size of the first tensor
- mode_1:mode_2:...:mode_N (product modes1): modes of the first tensor
- # of dimensions2: number of the dimensions of the second tensor
- dim_1:dim_2:...:dim_N (dimensions of tensor2): size of the second tensor
- mode_1:mode_2:...:mode_N (product modes2): modes of the second tensor
- input path: input path of the first and second tensors

- output path: output path of the product result
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.Tensor Operation

4.2.4.1. run_collapse.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_collapse.sh [input path] [output path] [collapsed mode] [# of reducers]

- input path: path of the input tensor
- output path: output path of the operation result
- collapsed mode: mode of the tensor to collapse
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.2. run_conv_sign.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_conv_sign [input path] [output path] [# of reducers]

- input path: path of the input tensor
- output path: output path of the operation result
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.3. run_conv_bin.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_conv_bin.sh [input path] [output path] [# of reducers]

- input path: path of the input tensor
- output path: output path of the operation result
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.4. run_find_cond.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_find_cond.sh [input path] [output path] [condition type] [scalar] [# of reducers]

- input path: path of the input tensor
- output path: output path of the operation result
- condition type: type of the comparison condition (e.g. ‘>’, ‘<’, ‘==’, ‘>=’, ‘<=’, ‘!=’)
- scalar: scalar value
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.5. run_find_elem.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_find_elem.sh [input path] [output path] [tensor index] [# of reducers]

- input path: path of the input tensor
- output path: output path of the operation result
- tensor index: index in the tensor
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.6. run_matricization.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_matricization.sh [input path] [output path] [dim_1:..:dim_N (tensor)] [matricized mode] [# of reducers]

- input path: path of the input tensor
- output path: output path of the operation result
- dim_1:..:dim_N (tensor): size of the input tensor
- matricized mode: mode of matricization
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.7. run_norm.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_norm.sh [input path] [output path] [x (l-x norm)]

- input path: path of the input tensor
- output path: output path of the operation result
- x (l-x norm): value x in the l-x norm (e.g. L-1 norm ‘1’, L-F norm ‘F’)

The output file is in the output path on HDFS.

4.2.4.8. run_permute_mode.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_permute_mode.sh [input path] [output path] [dim_1:...:dim_N (new order of dimension)]

- input path: path of the input tensor
- output path: output path of the operation result
- dim_1:...:dim_N (new order of dimension): new order of the modes of a tensor (e.g. 2:3:1 means that 2nd mode becomes 1st mode, 3rd mode becomes 2nd mode, and 1st mode becomes 3rd mode)

The output file is in the output path on HDFS.

4.2.4.9. run_scalar_op.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_scalar_op.sh [input path] [output path] [operation type] [scalar] [# of reducers]

- input path: path of the input tensor
- output path: output path of the operation result
- operation type: type of the scalar operation (e.g. ‘+’. ‘-‘, ‘*’, ‘/’)
- scalar: scalar value
- # of reducers: number of reducers to use

The output file is in the output path on HDFS.

4.2.4.10. run_scaling.sh

You need to upload to the HDFS directory a tensor stored in the sparse tensor format.

./run_scaling.sh [input path] [output path] [new mode length]

- input path: path of the input tensor
- output path: output path of the operation result
- new mode length: new mode length

The output file is in the output path on HDFS.