Fully Scalable Methods for Distributed Tensor Factorization

2016-04-10

Kijung Shin

1 General Information

- Version: 1.1
- Date: 04/10/2016
- Authors: Kijung Shin (kijungs@cs.cmu.edu), Lee Sael (sael@cs.stonybrook.edu), and U Kang (ukang@snu.ac.kr)

2 Introduction

This package implements CDTF and SALS, tensor factorization algorithms for high-order and large-scale data. It is fully written in Java and runs on Hadoop as well as on a single machine. The details of this package can be found in [1,2].

3 Installation

- This package requires the following software to be installed in the system and set in PATH.
  - Hadoop 1.0.3. or higher from http://hadoop.apache.org
  - Java 1.6.x. or higher, preferably from sun
- For compilation (optional), type *./compile.sh*
- For demo (optional), type *make*

4 PARAFAC model

4.1 Summary:

The entries of N-order tensor $\mathbf{X} \in (\mathbb{R}^{I_1 \times \ldots \times I_N})$ are approximated by the following formula:

$$x_{i_1 \ldots i_N} \approx \hat{x}_{i_1 \ldots i_N} = \sum_{k=1}^{K} \sum_{n=1}^{N} a_{i_n k}^{(n)}$$

where $a_{i_n k}^{(n)}$ is the (i_n, k)th element of $A^{(n)}$. Factor matrices, $A^{(1)}$ through $A^{(N)}$, are the result of the rank-K PARAFAC decomposition of $\mathbf{X}$, which minimizes the following loss function:

$$L(\mathrm{A}^{(1)}, \ldots, \mathrm{A}^{(\mathrm{N})}) = \sum_{(\mathrm{i}_1,\ldots,\mathrm{i}_\mathrm{N})\epsilon\Omega} (x_{i_1 \ldots i_N} - \sum_{k=1}^{K} \sum_{n=1}^{N} a_{i_n k}^{(n)})^2$$

where Ω is the set of $\mathbf{X}$'s observable entries.

For regularization, you can use L1 regularization, L2 regularization, or weighted-lambda regularization. Weighted-lambda-regularization is described in [3]. You also can add the non-negativity constraint so that factor matrices have non-negative entries.

4.2 Input

4.2.1 training data: $\mathbf{X}$'s entries used to calculate factor matrices.

- Format: $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, … , $[\mathrm{i}_\mathrm{N}]$, $[x_{i_1 \ldots i_N}]$ \n

Position	Type	Min	Max	Description
1	Integer	0	$\mathrm{I}_1 - 1$	i_1: 1st mode index
2	Integer	0	$\mathrm{I}_2 - 1$	i_2: 2nd mode index
…		...	…	…
N	Integer	0	$\mathrm{I}_\mathrm{N} - 1$	i_N: Nth mode index
N+1	Double			$x_{i_1 \ldots i_N}$: $(\mathrm{i}_1, \ldots, i_N)$th entry

- Example

  ► 2-dimensional data (2-order tensor): $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[x_{i_1 i_2}]$ \n

```
1,10,3.5

2,4,5.0

6,2,4.0
```

  ► 3-dimensional data (3-order tensor): $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[\mathrm{i}_3]$, $[x_{i_1 i_2 i_3}]$ \n

```
1,10,2 3.5

2,4,3,5.0

6,2,1,4.0
```

4.2.2 test data: $\mathbf{X}$'s entries used to measure the accuracy of estimation.

- Format: same with training data

4.2.3 query data: $\mathbf{X}$'s entries to be estimated

- Format: $[\mathrm{i}_1]$ $[\mathrm{i}_2]$ … $[\mathrm{i}_\mathrm{N}]$ \n

Position	Type	Min	Max	Description
1	Integer	0	$\mathrm{I}_1 - 1$	i_1: 1st mode index
2	Integer	0	$\mathrm{I}_2 - 1$	i_2: 2nd mode index
…		...	…	…

N	Integer	0	$I_N - 1$	i_N: Nth mode index

- Example

  ► 2-dimensional data (2-order tensor): $[i_1]$, $[i_2]$ \n

```
1,9

2,5

6,3
```

  ► 3-dimensional data (3-order tensor): $[i_1]$, $[i_2]$, $[i_3]$ \n

```
1,10,2,3.5

2,4,3,5.0

6,2,1,4.0
```

4.2.4 parameters

Name	Type	Min	Max	Description
training	String			Single version(**S**): path of training data on local disk Hadoop version(**H**): path of training data on HDFS
test	String			**S**: path of test data on local disk **H**: path of test data on HDFS
query	String			**S**: path of query data on local disk **H**: path of query data on HDFS
output	String			**S**: path to save outputs on local disk **H**: path to save outputs on HDFS
M	Integer	1		**S**: number of threads to use **H**: number of machines to use
Tout	Integer	1		number of outer iterations
N	Integer	1		dimension of data (order of a tensor)
K	Integer	1		rank
regularization	Binary	1	2	1: L1-regularization 2: L2-regularization
useWeight	Binary	0	1	1: weighted-lambda regularization, 0: no weight
nonNegative	Binary	0	1	1: non-negativity constraint, 0: no constraint
lambda	Double	0		regularization parameter for factor matrices
lambdaBias	Double	0		regularization parameter for bias terms
I_n	Integer	1		nth mode length
memory	Integer	1		amount of heap space (in MB) to allocate to each reducer

4.3 Output

4.3.1 performance.out: performance summary

- Format: [iteration] , [elapsed time] , [training RMSE] , [test RMSE] \n
- Example

```
iteration,elapsed_time,training_rmse,test_rmse

1,6779,0.900193,0.967152

2,11799,0.872561,0.943288

3,16373,0.860275,0.933825

4,20830,0.852764,0.928591

5,24828,0.847399,0.925174
```

4.3.2 estimate.out: estimated values for query data

- Format: $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, … , $[\mathrm{i}_\mathrm{N}]$, $[\hat{x}_{i_1 \dots i_N}]$ \n

Position	Type	Minimum	Maximum	Description
1	Integer	0	$\mathrm{I}_1 - 1$	i_1: 1st mode index
2	Integer	0	$\mathrm{I}_2 - 1$	i_2: 2nd mode index
…		...	…	…
N	Integer	0	$\mathrm{I}_\mathrm{N} - 1$	i_n: Nth mode index
N+1	Double			$\hat{x}_{i_1 \dots i_N}$: estimated $(\mathrm{i}_1, \dots, i_N)$th entry

- Example

  - 2-dimensional data (2-order tensor): $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[\hat{x}_{i_1 i_2}]$ \n

```
1,10,3.44

2,4,4.98

6,2,3.92
```

  - 3-dimensional data (3-order tensor): $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[\mathrm{i}_3]$, $[\hat{x}_{i_1 i_2 i_3}]$ \n

```
1,10,2,3.47

2,4,3,4.98

6,2,1,3.92
```

4.3.3 factor_matrices/n : $\mathrm{A}^{(\mathrm{n})}$ (n-th factor matrix)

- Format: $[\mathrm{i}_\mathrm{n}]$, $[a_{i_n 1}^{(n)}]$, $[a_{i_n 2}^{(n)}]$, … , $[a_{i_n K}^{(n)}]$ \n

Position	Type	Min	Max	Description
1	Integer	0	$\mathrm{I}_\mathrm{n} - 1$	i_n: row order
2	Double			$a_{i_n 1}^{(n)}$: 1st latent feature
3	Double			$a_{i_n 2}^{(n)}$: 2nd latent feature
…	…	…	…	….
K+1	Double			$a_{i_n K}^{(n)}$: Kth latent feature

- Example: $[\mathrm{i_n}]$, $[a^{(n)}_{i_n1}]$, $[a^{(n)}_{i_n2}]$, $[a^{(n)}_{i_n3}]$, $[a^{(n)}_{i_n4}]$, $[a^{(n)}_{i_n5}]$ \n

```
0,0.531,0.422,0.234,0.161,0.231

1,0.223,0.491,0.481,0.592,0.351

2,0.334,0.478,0.123,0.439,0.692
```

4.4 Algorithms

4.4.1 CDTF (Coordinate Descent for Tensor Factorization) [1,2]

- How to run
  - Single machine version

```
./run_cdtf_single.sh [training] [output] [M] [Tout] [Tin] [N] [K] [regularization] [useWeight] [nonNegative] [lambda] [I_1] [I_2] ... [I_N] [test] [query]
```

    - [Tin]: number of inner iterations
    - [test] and [query] are optional

  - Hadoop version

```
./run_cdtf_hadoop.sh [training] [output] [M] [Tout] [Tin] [N] [K] [regularization] [useWeight] [nonNegative] [lambda] [I_1] [I_2] ... [I_N] [memory] [test] [query]
```

    - [Tin]: number of inner iterations
    - [test] and [query] are optional

4.4.2 SALS (Subset Alternating Least Square) [1,2]

- How to run
  - Single machine version

```
./run_sals_single.sh [training] [output] [M] [Tout] [Tin] [N] [K] [C] [useWeight] [lambda] [I_1] [I_2] ... [I_N] [test] [query]
```

    - [Tin]: number of inner iterations
    - [C]: number of parameters updated at a time
    - [useWeight]: 1: weighted-lambda-regularization, 0: L2-regularization
    - [test] and [query] are optional

  - Hadoop version

```
./run_sals_hadoop.sh [training] [output] [M] [Tout] [Tin] [N] [K] [C] [useWeight] [lambda]
```

```
[I_1] [I_2] ... [I_N] [memory] [test] [query]
```

- [Tin]: number of inner iterations
- [C]: number of parameters updated at a time
- [useWeight]: 1: weighted-lambda-regularization, 0: L2-regularization
- [test] and [query] are optional

5 Bias Model

5.1 Summary

The entries of N-order tensor data $\mathbf{X} \in (\mathbb{R}^{I_1 \times \ldots \times I_N})$ are approximated by the following formula:

$$x_{i_1 \ldots i_N} \approx \hat{x}_{i_1 \ldots i_N} = \mu + \sum_{n=1}^{N} b_{i_n}^{(n)} + \sum_{k=1}^{K} \sum_{n=1}^{N} a_{i_n k}^{(n)}$$

where μ is the average of the observable entries of $\mathbf{X}$, $b_{i_n}^{(n)}$ is the $\mathrm{i_n}$th entry of $\mathrm{b}^{(\mathrm{n})}$, and $a_{i_n k}^{(n)}$ is the $(\mathrm{i_n}, \mathrm{k})$th entry of $\mathrm{A}^{(\mathrm{n})}$. Bias terms, $\mathrm{b}^{(1)}$ through $\mathrm{b}^{(\mathrm{N})}$, and factor matrices, $\mathrm{A}^{(1)}$ through $\mathrm{A}^{(\mathrm{N})}$, are set to values minimizing the following loss function:

$$L\left(\mathrm{b}^{(1)}, \ldots, \mathrm{b}^{(\mathrm{N})}, \mathrm{A}^{(1)}, \ldots, \mathrm{A}^{(\mathrm{N})}\right) = \sum_{(\mathrm{i}_1, \ldots, \mathrm{i}_\mathrm{N}) \epsilon \Omega} (x_{i_1 \ldots i_N} - \mu - \sum_{n=1}^{N} b_{i_n}^{(n)} - \sum_{k=1}^{K} \sum_{n=1}^{N} a_{i_n k}^{(n)})^2$$

where Ω is the set of $\mathbf{X}$'s observable entries.

For regularization, you can use L1 regularization, L2 regularization, or weighted-lambda regularization. Weighted-lambda-regularization is described in [3]. You also can add the non-negativity constraint so that bias terms and the entries of factor matrices have non-negative values.

5.2 Input

5.2.1 training data: see 4.2.1.

5.2.2 test data: see 4.2.2.

5.2.3 query data: see 4.2.3.

5.2.4 parameters: see 4.2.4.

5.3 Output

5.3.1 performance.out: performance summary: see 4.3.1.

5.3.2 estimate.out: estimated values for query data: see 4.3.2.

5.3.3 factor_matrices/n : $\mathrm{A}^{(\mathrm{n})}$ (n-th factor matrix): see 4.3.3.

5.3.4 biased_terms/n : $b^{(\mathrm{n})}$ (n-th bias vector)

- Format: $[\mathrm{i_n}]$, $[b_{i_n}^{(n)}]$ \n

Position	Type	Min	Max	Description
1	Integer	0	$\mathrm{I_n} - 1$	$\mathrm{i_n}$: row order
2	Double			$b_{i_n}^{(n)}$: bias term

- Example: $[\mathrm{i_n}]$, $[b_{i_n}^{(n)}]$ \n

```
0,0.431

1,-0.128

2,0.364
```

5.3.5 mu : μ the average of the observable entries of $\mathbf{X}$

- Format: $[\mu]$

Position	Type	Min	Max	Description
1	Double			μ: the average of the observable entries of $\mathbf{X}$

- Example: $[\mu]$

```
3.234
```

5.4 Algorithms

5.4.1 Bias-CDTF (Bias Coordinate Descent for Tensor Factorization) [1]

- How to run
  - ► Single machine version

```
./run_cdtf_bias_single.sh [training] [output] [M] [Tout] [Tin] [N] [K] [regularization]
[useWeight] [nonNegative] [lambda] [lambdaBias] [I_1] [I_2] ... [I_N] [test] [query]
```

    - ➢ [Tin]: number of inner iterations
    - ➢ [test] and [query] are optional
  - ► Hadoop version

```
./run_cdtf_bias_hadoop.sh [training] [output] [M] [Tout] [Tin] [N] [K] [regularization]
[useWeight] [nonNegative] [lambda] [lambdaBias] [I_1] [I_2] ... [I_N] [memory] [test]
[query]
```

    - ➢ [Tin]: number of inner iterations
    - ➢ [test] and [query] are optional

5.4.2 Bias-SALS (Bias Subset Alternating Least Square) [1]

- How to run
  - Single machine version

```
./run_sals_bias_single.sh [training] [output] [M] [Tout] [Tin] [N] [K] [C] [useWeight]
[lambda] [lambdaBias] [I_1] [I_2] ... [I_N] [test] [query]
```

    - [Tin]: number of inner iterations
    - [useWeight]: 1: weighted-lambda-regularization, 0: L2-regularization
    - [test] and [query] are optional

  - Hadoop version

```
./run_sals_bias_hadoop.sh [training] [output] [M] [Tout] [Tin] [N] [K] [C] [useWeight]
[lambda] [lambdaBias] [I_1] [I_2] ... [I_N] [memory] [test] [query]
```

    - [Tin]: number of inner iterations
    - [useWeight]: 1: weighted-lambda-regularization, 0: L2-regularization
    - [test] and [query] are optional

6 Coupled Model

6.1 Summary

The entries of N_x-order tensor data $\mathbf{X} \in (\mathbb{R}^{\mathrm{I}_1 \times \ldots \times I_{\mathrm{N}_x}})$ and N_y-order tensor data $\mathbf{Y} \in (\mathbb{R}^{\mathrm{I}_1 \times \ldots \times I_{\mathrm{N}_y}})$ are approximated by the following formulas:

$$\hat{x}_{i_1 \ldots i_{\mathrm{N}_x}} = \sum_{k=1}^{K} \sum_{n=1}^{\mathrm{N}_x} {}_x a_{i_n k}^{(n)}, \qquad \hat{y}_{i_1 \ldots i_{\mathrm{N}_y}} = \sum_{k=1}^{K} \sum_{n=1}^{\mathrm{N}_y} {}_y a_{i_n k}^{(n)}$$

where ${}_x a_{i_n k}^{(n)}$ is the $(\mathrm{i_n}, \mathrm{k})$th element of ${}_x\mathrm{A}^{(\mathrm{n})}$, and ${}_y a_{i_n k}^{(n)}$ is the $(\mathrm{i_n}, \mathrm{k})$th element of ${}_y\mathrm{A}^{(\mathrm{n})}$. Factor matrices, ${}_x\mathrm{A}^{(1)}$ through ${}_x\mathrm{A}^{(\mathrm{N}_x)}$ and ${}_y\mathrm{A}^{(1)}$ through ${}_y\mathrm{A}^{(\mathrm{N}_y)}$, are set to values minimizing the following loss function under the condition that ${}_x\mathrm{A}^{(1)} = {}_y\mathrm{A}^{(1)}$:

$$L({}_x\mathrm{A}^{(1)}, \ldots, {}_x\mathrm{A}^{(\mathrm{N}_x)}, {}_y\mathrm{A}^{(1)}, \ldots, {}_y\mathrm{A}^{(\mathrm{N}_y)})$$

$$= \sum_{(\mathrm{i}_1,\ldots,\mathrm{i}_\mathrm{N}) \in \Omega_x} (x_{i_1 \ldots i_N} - \sum_{k=1}^{K} \sum_{n=1}^{N_x} {}_x a_{i_n k}^{(n)})^2 + \sum_{(\mathrm{i}_1,\ldots,\mathrm{i}_\mathrm{N}) \in \Omega_y} (y_{i_1 \ldots i_N} - \sum_{k=1}^{K} \sum_{n=1}^{N_y} {}_y a_{i_n k}^{(n)})^2$$

where Ω_x is the set of $\mathbf{X}$'s observable entries, and Ω_y is the set of $\mathbf{Y}$'s observable entries

For regularization, you can use either weighted-lambda-regularization or L2 regularization. Weighted-lambda-regularization is described in [3].

6.2 Input

6.2.1 training data: see 4.2.1.

6.2.2 coupled-tensor data: $\mathbf{Y}$'s entries.

- Format: $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, … , $[\mathrm{i}_{\mathrm{N}_y}]$, $[y_{i_1 \dots i_N}]$ \n

Position	Type	Min	Max	Description
1	Integer	0	$\mathrm{I}_1 - 1$	i_1: 1st mode index
2	Integer	0	$\mathrm{I}_2 - 1$	i_2: 2nd mode index
…		...	…	…
N	Integer	0	$\mathrm{I}_\mathrm{N} - 1$	$\mathrm{i}_{\mathrm{N}_y}$: N_yth mode index
N+1	Double			$y_{i_1 \dots i_{\mathrm{N}_y}}$: $\left(\mathrm{i}_1, \dots, i_{N_y}\right)$th entry of $\mathbf{Y}$

- Example

  ► 2-dimensional data (2-order tensor): $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[y_{i_1 i_2}]$ \n

```
1,10,3.5

2,4,5.0

6,2,4.0
```

  ► 3-dimensional data (3-order tensor): $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[\mathrm{i}_3]$, $[y_{i_1 i_2 i_3}]$ \n

```
1,10,2 3.5

2,4,3,5.0

6,2,1,4.0
```

6.2.3 test data: see 4.2.2.

6.2.4 query data: see 4.2.3.

6.2.5 parameters: see 4.2.4 for the rest parameters

Name	Type	Min	Max	Description
coupled_tensor	String			path of coupled-tensor data on local disk
Nx	Integer	1		dimension of data (order of an input tensor)
Ny	Integer	1		order of a coupled tensor
I_n	Integer	1		nth mode length of an input tensor
J_n	integer	1		nth mode length of a coupled tensor

6.3 Output

6.3.1 performance.out: performance summary: see 4.3.1.

6.3.2 estimate.out: estimated values for query data: see 4.3.2.

6.3.3 factor_matrices/n : $\mathrm{A}^{(\mathrm{n})}$ (n-th factor matrix): see 4.3.3.

6.4 Algorithms

6.4.1 Coupled-CDTF (Coupled Coordinate Descent for Tensor Factorization) [1]

- How to run
  - ► Single machine version

```
./run_cdtf_coupled_single.sh [training] [coupled_tensor] [output] [M] [Tout] [Tin] [Nx]
[Ny] [K] [useWeight] [lambda] [I_1] [I_2] ... [I_Nx] [J_1] [J_2] … [J_Ny] [test] [query]
```

    - [Tin]: number of inner iterations
    - [useWeight]: 1: weighted-lambda-regularization, 0: L2-regularization
    - [test] and [query] are optional

6.4.2 Coupled-SALS (Coupled Subset Alternating Least Square) [1]

- How to run
  - ► Single machine version

```
./run_sals_coupled_single.sh [training] [coupled_tensor] [output] [M] [Tout] [Tin] [Nx]
[Ny] [K] [C] [useWeight] [lambda] [I_1] [I_2] ... [I_Nx] [J_1] [J_2] … [J_Ny] [test] [query]
```

    - [Tin]: number of inner iterations
    - [useWeight]: 1: weighted-lambda-regularization, 0: L2-regularization
    - [test] and [query] are optional

7 Reference

[1] Kijung Shin, Lee Sael, and U Kang, “Fully Scalable Methods for Distributed Tensor Factorization” (Submitted)

[2] Kijung Shin and U Kang, “Distributed Methods for High-dimensional and Large-scale Tensor Factorization”. IEEE International Conference on Data Mining (ICDM), 2014.

[3] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. “Large-scale parallel collaborative filtering for the netflix prize”. In Algorithmic Aspects in Information and Management, pages 337–348. Springer, 2008.