Distributed Methods for High-dimensional and Large-scale Tensor Factorization

2014-09-29

Kijung Shin

1 General Information

- Version: 1.0
- Date: 09/29/2014
- Authors: Kijung Shin (koreaskj@snu.ac.kr), U Kang (ukang@cs.kaist.ac.kr)

2 Introduction

This package implements SALS and CDTF, tensor factorization algorithms for high-dimensional 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 the following paper:

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

3 Installation

This package requires the following software to be installed in the system and set in PATH.

- Hadoop 1.0.3. or greater from http://hadoop.apache.org
- Java 1.6.x. or greater, preferably from sun

4 Tensor factorization

4.1 Model:

The entries of N-dimensional tensor $\mathbf{X} \in (\mathbb{R}^{\mathrm{I}_1 \times \ldots \times I_N})$ 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 $(\mathrm{i_n}, \mathrm{k})$th element of $\mathrm{A}^{(\mathrm{n})}$. Factor matrices, $\mathrm{A}^{(1)}$ through $\mathrm{A}^{(\mathrm{n})}$, are the result of rank

K PARAFAC decomposition of $\mathbf{X}$, which minimizes following loss function:

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

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

For regularization, you can use either weighted-lambda-regularization or L2 regularization. Weighted-lambda-regularization is described in the following paper:

- 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.

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: $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[x_{i_1 i_2}]$ \n

```
1,10,3.5
2,4,5.0
6,2,4.0
```

  - ► 3-dimensional data: $[\mathrm{i}_1]$, $[\mathrm{i}_2]$, $[\mathrm{i}_3]$, $[x_{i_1 i_2 i_3}]$ \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	$I_1 - 1$	i_1: 1st mode index
2	Integer	0	$I_2 - 1$	i_2: 2nd mode index
…		...	…	…
N	Integer	0	$I_N - 1$	i_n: Nth mode index

- Example

  ► 2-dimensional data: $[i_1]$, $[i_2]$ \n

```
1,9

2,5

6,3
```

  ► 3-dimensional data: $[i_1]$, $[i_2]$, $[i_3]$ \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
K	Integer	1		rank
lambda	double	0	1	regularization parameter
useWeight	Integer			1 : weighted lambda regularization, 0: L2 regularization
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: $[i_1]$, $[i_2]$, … , $[i_N]$, $[\hat{x}_{i_1 \ldots i_N}]$ \n

Position	Type	Minimum	Maximum	Description
1	Integer	0	$I_1 - 1$	i_1: 1st mode index
2	Integer	0	$I_2 - 1$	i_2: 2nd mode index
…		...	…	…
N	Integer	0	$I_N - 1$	i_n: Nth mode index
N+1	Double			$\hat{x}_{i_1 \ldots i_N}$: estimated $(i_1, \ldots, i_n)$th entry

- Example
  - ► 2-dimensional data: $[i_1]$, $[i_2]$, $[\hat{x}_{i_1 i_2}]$ \n

```
1,10,3.44
2,4,4.98
6,2,3.92
```

  - ► 3-dimensional data: $[i_1]$, $[i_2]$, $[i_3]$, $[\hat{x}_{i_1 i_2 i_3}]$ \n

```
1,10,2,3.47
2,4,3,4.98
6,2,1,3.92
```

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

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

Position	Type	Min	Max	Description
1	Integer	0	$I_n - 1$	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: [i_n] , [$a^{(n)}_{i_n 1}$] , [$a^{(n)}_{i_n 2}$] , [$a^{(n)}_{i_n 3}$] , [$a^{(n)}_{i_n 4}$] , [$a^{(n)}_{i_n 5}$] \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)

- How to run
  - ► Single machine version

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

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

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

    - ➢ [Tin]: number of inner iterations
    - ➢ [test] and [query] are optional
- Reference
  - ► Kijung Shin, U Kang, Distributed Methods for High-dimensional and Large-scale Tensor Factorization. IEEE International Conference on Data Mining(ICDM), December 2014.
  - ► H.-F. Yu, C.-J. Hsieh, S. Si, I. S. Dhillon, Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems. IEEE International Conference on Data Mining(ICDM), December 2012.

4.4.2 SALS (Subset Alternating Least Square)

- How to run
  - ► Single machine version

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

    - ➢ [Tin]: number of inner iterations
    - ➢ [C]: number of parameters updated at a time

- [test] and [query] are optional

► Hadoop version

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

- [Tin]: number of inner iterations
- [C]: number of parameters updated at a time
- [test] and [query] are optional

● Reference

► Kijung Shin, U Kang, Distributed Methods for High-dimensional and Large-scale Tensor Factorization. IEEE International Conference on Data Mining(ICDM), December 2014.