Scalable Principal Component Analysis Implementation on Husky

Liu Jie, Bao Ergute

Dept. of Compuer Science and Engineering, The Chinese University of Hong Kong

Introduction

This study aims to implement **scalable PCA** algorithm based on probabilistic PCA on a distributed computing platform, **Husky**, which is developed by Prof. James Cheng's team, and employ several optimizations [1] to support large datasets.

Motivation

- Nowadays, websites, social networks and sensors generate massive amount of data with possibly millions of features.
- Current libraries that offer PCA for distributed clusters, such as Mahout on **MapReduce** and MLlib on **Spark**, do not scale well to support big data analysis.

Basic Concepts

- **PCA** : Principal Component Analysis, a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.
- $\mathbf{y}$: input matrix of size $N \times D$, that is, N samples or observations, D features or variables.
- $\mathbf{X}$: principal components of size $N \times d$, where d is the number of components to be kept. Also the latent variable in probabilistic model.
- $\mathbf{C}$: transformation matrix of size $D \times d$

Method

- Probabilistic PCA
- PCA is presented as a latent variable model that seeks a linear relation between a D-dimensional observed data vector $\mathbf{y}$ and a d-dimensional latent variable $\mathbf{x}$.

$$\mathbf{y} = \mathbf{C} * \mathbf{x} + \mu + \varepsilon$$

Given N observations $\{\mathbf{y}\}_1^N$ as the input data, the log likelihood is given by:

$$L(\{\mathbf{y}\}) = \sum_{r=1}^{N} \ln\{p(\mathbf{y})\}$$

.

MLE of C is obtained by optimizing $arg_C max L(\{\mathbf{y}\})$.
The main idea of probabilistic PCA is that the MLE solution is the solution of PCA. *Expectation Maximization* algorithm is adopted to solve MLE problem. This is the framework of our PCA algorithm.

References

[1] T. Elgamal, M. Yabandeh, A. Aboulnaga, W. Mustafa, and M. Hefeeda. spca: Scalable principal component analysis for big data on distributed platforms. In SIGMOD, pages 79–91, 2015.

[2] F. Yang, J. Li, and J. Cheng. Husky: Towards a more efficient and expressive distributed computing framework. PVLDB, 9(5):420–431, 2016.

Optimizations

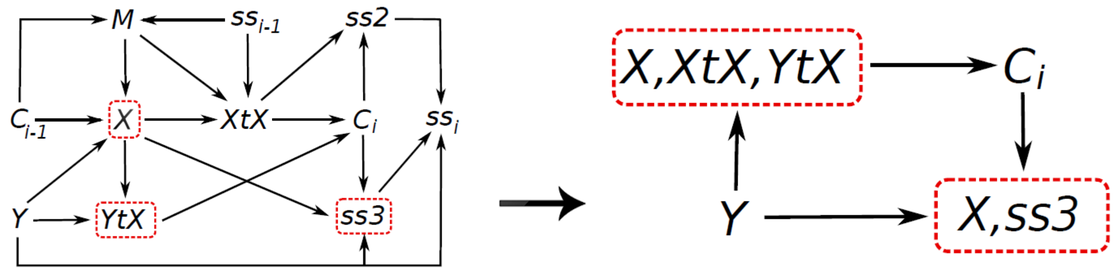

Figura: From PPCA to sPCA

- Propagate mean to leverage sparsity, and keep original matrix Y and mean vector Ym in two separate data structures.

$$Y_C * C = (Y - Ym) * C = Y * C - Ym * C$$

- Minimize the intermediate data. Store vectors and small matrices locally and trade intermediate data footprint with redundant computation. The above figure shows the change of job scheduling.
- Make matrix multiplication more efficient.

$$(A_T * B) = \sum_{i=1}^{D} (A_i)^T * B_i$$

If instead A * B we want to compute $A^T * B$, then we can use the formula above to reduce random accesses to the matrices.

Performance Evaluation

- Our experiments show that sPCA outperforms MLlib-PCA in terms of running time and intermediate data generated during the computation when processing datasets with $O(1k)$ dimensions or more.

Input($N \times D$)	d	SPCA	MLlib-PCA
353 × 2K	300	80sec	225sec
353 × 2K	100	22sec	170sec
353 × 2K	10	6sec	120sec
64 × 47236	50	315sec	fail
20242 × 47236	10	100sec/iteration	fail

Conclusion

We analyzed different methods for computing PCA, and implemented scalable PCA (sPCA) algorithm which is based on probabilistic PCA on Husky platform. While PCA on popular platforms (MapReduce, Spark) cannot scale to large datasets due to computation and communication bottlenecks, sPCA has been proved to outperform PCA on Spark and be capable of processing datasets with **O(100K)** dimensions.

Acknowledgement

We would like to thank Professor James Cheng and all phd mentors for their great assistance.