On Device Federated PCA & Subspace Tracking

Andreas Grammenos*
University of Cambridge
Cambridge, UK
ag926@cl.cam.ac.uk

Cecilia Mascolo
University of Cambridge
Cambridge, UK
cm542@cl.cam.ac.uk

Jon Crowcroft*
University of Cambridge
Cambridge, UK
jac22@cl.cam.ac.uk

MOTIVATION

In recent years there has been an increasing desire to perform traditional machine learning and inference tasks such as feature extraction and data summarisation *on-device*. Further, the data used for training are usually sourced from the each individual device which can be viewed as d-dimensional vectors collected in a streaming fashion. Notably, this can be a challenging task when operating in a resource constrained environment such as mobile or IoT devices. Concretely, d is even moderately large, most devices will not be able to store the collected data, nor compute algorithms normally requiring at least $\Omega(d^2)$ storage such as Subspace Tracking. Further, due to privacy and performance considerations it is highly desirable to be able to perform these tasks on device. In this work we try to tackle the aforementioned problem by introducing a technique which can be used to summarise each subspace locally *on-device* and then efficiently merge them globally.

PROBLEM STATEMENT

The increasing need of being able to use mobile and IoT devices to analyze high-dimensional, heterogeneous data and extract useful insights *on-device* is becoming more prevalent than ever before. When tackling the task of extracting information from data out of many tools available arguably the most commonly used ones are Principal Component Analysis (PCA) [4, 6] and Subspace Tracking [7, 8]. These techniques are frequently used for discovering a linear structure or reducing dimensionality in data, so they have become an essential component in inference, machine-learning, and data-science pipelines. In a nutshell, given a matrix $\mathbf{Y} \in \mathbb{R}^{d \times n}$ of n feature vectors of dimension d, these techniques aim to build a low-dimensional subspace of $\mathbb{R}^d$ that captures the directions of maximum variance in the data contained in $\mathbf{Y}$. This is useful for a number of reasons, for example, to reduce the dimensionality of a high-dimensional dataset in order to minimise the computational cost of other expensive operations.

Their computation is directly related to the Singular Value Decomposition (SVD) [1, 5] which can decompose any matrix into a linear combination of orthonormal rank-1 matrices weighted by positive scalars. Naturally, the computation of Subspaces, PCA, and SVD has been studied for decades, but with the recent rise of the need to perform *on-device* inference, many new and diverse challenges arise. This is primarily due to the mismatch between the limited available resources and the size of the data to be processed. In the context of high-dimensional data, the main limitation stems from the fact that, in the absence of structure, performing SVD on a matrix $\mathbf{Y} \in \mathbb{R}^{d \times n}$ requires $O(d^2 n + d^3)$ computation time and $O(d^2)$ memory.

*Author is also affiliated with The Alan Turing Institute

FEDERATED COMPUTATION

To be able to fully leverage a federated computation environment mainly comprised out of mobile or IoT devices, we would like also to be able to perform the computation in an incremental, streaming fashion. Indeed, there have been some previous works on how to perform incremental SVD such as in [3]; yet, note that this particular scheme is only incremental and *not* streaming. This means that every result has to be computed in-full in order to be processed, merged, and propagated for the user to get a final result, so is not fit for a federated computing approach. This can be problematic in many cases, as various applications (e.g. stock exchange tracking, video frame classification, image recognition) require to have intermediate or *partial* solutions before the whole dataset is processed. However, this scheme can be combined with a streaming algorithm such as the one presented in [2] capable of producing streaming intermediate results that can be merged earlier on in the update propagation.

Adaptive Rank Estimation

Further when operating in a such an environment it would be convenient to have schemes where each individual device can adjust, independently of each other, their rank estimate based on the distribution of the data seen so far. This is because we expect to have nodes which observe different distributions and likely will require, over time, to adjust the number of principal components kept in order to accurately track the data distribution. To this end, and in order to devise a simple, yet efficient adaptive mechanism for selecting the estimated matrix rank of the fly, we propose a regularisation scheme based solely on the current singular values estimate and their contribution to the total approximation discovered so far.

ONGOING WORK

Currently our work is focusing on finalising the implementation on actual mobile devices and perform large scale tests benchmarking the aforementioned method in a real-world scenario.

REFERENCES

[1] C. Eckart and G. Young. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* 1 (1936), 211–218. https://doi.org/10.1007/BF02288367
[2] Armin Eftekhari, Raphael A Hauser, and Andreas Grammenos. 2018. MOSES: A Streaming Algorithm for Linear Dimensionality Reduction. *arXiv preprint arXiv:1806.01304* (2018).
[3] MA Iwen and BW Ong. 2016. A distributed and incremental svd algorithm for agglomerative data analysis on large networks. *SIAM J. Matrix Anal. Appl.* 37, 4 (2016), 1699–1718.
[4] Ian Jolliffe. 2011. Principal component analysis. In *International encyclopedia of statistical science*. Springer, 1094–1096.
[5] L. Mirsky. 1966. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford* (1966), 1156–1159.
[6] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2, 11 (1901), 559–572.

[7] Gilbert W Stewart. 1992. An updating algorithm for subspace tracking. *IEEE Transactions on Signal Processing* 40, 6 (1992), 1535–1541.
[8] Bin Yang. 1995. Projection approximation subspace tracking. *IEEE Transactions on Signal processing* 43, 1 (1995), 95–107.