Flexible Transformations for Learning Big Data

Azalia Mirhoseini Ebrahim M. Songhori Bita Darvish Rouhani
Farinaz Koushanfar

azalia, ebrahim, bita, farinaz@rice.edu

Electrical and Computer Engineering, Rice University, Houston TX 77005, USA

ABSTRACT

This paper proposes a domain-specific solution for iterative learning of big and *dense (non-sparse)* datasets. A large host of learning algorithms, including linear and regularized regression techniques, rely on iterative updates on the data connectivity matrix in order to converge to a solution. The performance of such algorithms often severely degrade when it comes to large and dense data. Massive dense datasets not only induce obligatory large number of arithmetics, but they also incur unwanted message passing cost across the processing nodes. Our key observation is that despite the seemingly dense structures, in many applications, data can be transformed into a new space where sparse structures become revealed. We propose a scalable data transformation scheme that enables creating versatile sparse representations of the data. The transformation can be tuned to benefit the underlying platform's cost and constraints. Our evaluations demonstrate significant improvement in energy usage, runtime, and memory footprint, within guaranteed user-defined error bounds.

Categories and Subject Descriptors

I.1.2 [**Computing Methodologies**]: Symbolic and algebraic manipulation—*Algorithms*

Keywords

Big and dense data, Sparse factorization, Subspace sampling, Performance optimization

1. INTRODUCTION

Since finding a direct solution to a matrix-based analysis problem requires the (unscalable) matrix inversion, most contemporary data analysis methods use iterative computing to converge to a solution [3]. The major bottleneck of iterative algorithms is the costly operations on the correlation matrix. Let us denote the *data matrix* by $\mathbf{A} \in \mathcal{R}^{m\times n}$, where n is the total number of samples and m is the number of features per sample. We denote the *correlation matrix* by $\mathbf{G} \in \mathcal{R}^{n\times n}$, where $\mathbf{G} = \mathbf{A}^T\mathbf{A}$. At each iteration t, the following process is applied to update a *solution vector*, $\mathbf{x}$: $\mathbf{x}^{t+1} = \Phi(\mathbf{G}\mathbf{x}^t, \mathbf{x}^t)$, where Φ is a lightweight operator determined by the algorithm. The update is applied iteratively until convergence is achieved.

The complexity of the update arises from computing the matrix-vector product $\mathbf{G}\mathbf{x}^t$ at each step. Computing and storing $\mathbf{G}$ for massive datasets with dense correlations become infeasible as it requires $\mathcal{O}(n^3)$ arithmetic operations and $\mathcal{O}(n^2)$ storage. To circumvent this, only $\mathbf{A}$ is stored. Thus, the $\mathbf{G}\mathbf{x}^t$ computation is instead done using $\mathbf{A}^T\mathbf{A}\mathbf{x}^t$. In settings where $\mathbf{A}$ has to be partitioned across distributed processing nodes, computing the correlations between data samples that reside on different nodes incur communication overhead.

Several classes of fast-growing data, including the image and video content, contain dense (non-sparse) dependencies, i.e., a large number of non-zeros in the data correlation matrix. While available approaches such as [5] rely sparsity of the data connectivity matrix for efficient partitioning, when data exhibits dense dependencies, it is inherently impossible to find efficient cuts.

This work aims to address the challenges that arise due to the large number of arithmetic and message passing operations required for executing iterative algorithms on massive and densely correlated data. We propose a tunable data transformation scheme that enables producing versatile sparse representations of the same dataset, within a user-defined approximation error. The degree of freedom in transformation can be leveraged to optimize the performance cost of iterative updates in terms of energy usage, runtime, and memory on the pertinent platform.

2. A FLEXIBLE AND SPARSIFYING DATA TRANSFORMATION

Here we present our parametric data transformation methodology. Our objective is to find an approximate solution to the following NP hard problem:

$$\min_{L,\mathbf{B}_{m\times L},\mathbf{C}_{L\times n}} \|\mathbf{c}\|_\mathbf{0} \ \text{ s.t. } \ \|\mathbf{a_i} - \mathbf{B}\mathbf{c_i}\|_\mathbf{F} \leq \delta, \ \textbf{for } \mathbf{1} \leq \mathbf{i} \leq \mathbf{n}. \tag{1}$$

where $\mathbf{a}_i$'s and $\mathbf{c_i}$'s are columns of $\mathbf{A}$ and $\mathbf{C}$, $\|\cdot\|_0$ measures the total number of non-zeros, $\|\cdot\|_F$ measures the Frobenius norm, and δ is an approximation error. Equation 1 naturally expresses our goal to find sparse transformations of the dense data.

SIGMETRICS'15, June 15-19, 2015, Portland, OR, USA.
ACM 978-1-4503-3486-0/15/06.
http://dx.doi.org/10.1145/2745844.2745889.

Algorithm 1 : Parametric sparsifying data transformation

```
Input: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, error tolerance $\delta$, and the maximum number of columns to select $L$.
Output: A sparse matrix $\mathbf{C} \in \mathbb{R}^{L \times n}$ and a dense matrix $\mathbf{B} \in \mathbb{R}^{m \times L}$ such that $\|\mathbf{a}_i - \mathbf{B}\mathbf{c_i}\|_\mathbf{F} \leq \delta$ for each $1 \leq i \leq n$.

1. Update B by selecting $L$ columns uniformly at random from A
2. Normalize columns of B: $\|b_i\|_2 = 1$.
3. For each $1 \leq i \leq n$, compute $\mathbf{c_i}$ by applying OMP to solve Equation 1 within error bound $\delta$: $\mathbf{a}_i - B\mathbf{c_i} \leq \delta$.
```

Our approach to solve Equation 1 consists of 2 steps. First matrix **B** is created by using a random Column-Subset-Selection (CSS) approach; a set of L columns of **A** are selected at random to create **B**. L should be large-enough such that the error criteria is met. We refer to the general results on random CSS, provided in [4], for bounds on L for a given error. Oncer **B** is formed, Equation 1 becomes equivalent to a sparse approximation problem. Each column $\mathbf{c_i}$ is a sparse approximation of the corresponding column $\mathbf{a}_i$, with respect to **B** and the user-specified approximation error δ. We solve this problem using an efficient greedy routine called *Orthogonal Matching Pursuit* (OMP) [2].

Sparsity guarantees for C. When a dataset lives in a union of lower dimensional subspaces, as it is the case in many large-scale scenarios, each data point corresponds to a superposition of a few points from its own subspace [6]. More precisely, let us assume that **A** lives in a union of s subspaces of $\mathbb{R}^n$, where each subspace is k_i-dimensional $(1 \leq i \leq s)$. Then the columns of **A** that belong to each subspace i will be represented by at most k_i non-zeros in **c**. Note that the low-rank model is an instance of the union-of-subspace model where $s = 1$. In Algorithm 1, matrix **B** is the collection of all subspaces that **A** lives in. The OMP routine greedily approximates each $\mathbf{a}_i$, as a superposition of few columns of **B** that best represent $\mathbf{a}_i$.

Platform-aware tuning of transformation . Once the transformation is completed, subsequent iterative analysis on $\mathbf{A}^T\mathbf{A}$ are replaced by $(\mathbf{BC})^T\mathbf{BC}$. The sparsity of **C** enables efficient partitioning and computation. One can exploit the existing work in the field of high performance computing to optimally distribute and apply the updates on matrices **B** and **C**. In particular, if n is much larger than both m and the number of columns in **x**, as it is the case in many large-scale learning problems, the number of communicated words increases linearly with L [1].

On the other hand, increasing parameter L results in sparser transformations. The larger sparsity is achieved due to the higher redundancy in **B**. Thus, there is a trade-off between the number of non-zeros (or the cost of computation and the memory footprint) of the transformed data and the communication overhead. We can leverage this property to adaptively tune the transformation for the target platform.

3. EVALUATIONS

We performed our evaluations on IBM iDataplex computer cluster. Our test datasets include a database of video frames (size 1764×100000) and a database of light-field image patches (size 18496×100000).

Table 1: Performance improvement in terms of number of floating point operations per iteration and memory usage.

Method	Video	Light field
Ours (GFLOP-per-iteration)	0.061	0.068
Baseline (GFLOP-per-iteration)	0.227	3.69
Ours (Memory MB)	244.64	275.71
Baseline (Memory MB)	1280.00	14797.96

Tunable sparse representations. We tested Algorithm 1 on light field data by setting $\delta = 0.1$. Figure 1 shows the average number of non-zeros per column of C as L increases. At first the number of non-zeros increases with L until the transformation error criteria is met. Afterwards, increasing L further sparsifies **C**. Since L governs the communication cost, depending on the relative cost of arithmetics and communication on a platform, optimal L can be chosen.

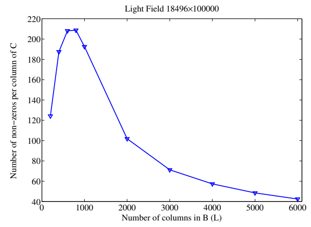

Figure 1: By varying the number of columns in **B** in our transformation, various sparsity levels in coefficient matrix **C** can be achieved. This property can be used to tune data representation w.r.t costs of the pertinent platform.

Energy and memory efficiency. Table 1 provides the cost in terms of number of floating point operations (FLOPs) per iteration using our approach i.e., $(\mathbf{BC})^T\mathbf{BCx}$ vs. the baseline, i.e., $\mathbf{A}^T\mathbf{Ax}$. It also provides the cost in terms of memory usage using our approach (memory for storing **B** and **C**) vs. the baseline (memory for storing **A**). The error is set to $\delta = 0.1$. Since energy usage is highly dependent on the number of FLOPs, drastic reductions in the number of FLOPs (e.g., more than 54 times for Light Field data) is an indication of significant energy performance improvement.

4. REFERENCES

[1] G. Ballard, A. Buluc, J. Demmel, L. Grigori, B. Lipshitz, O. Schwartz, and S. Toledo. Communication optimal parallel multiplication of sparse random matrices. SPAA '13, pages 222–231.

[2] G. Davis, S. Mallat, and Z. Zhang. Adaptive time-frequency decompositions. *Opt. Engin SPIE*, pages 2183–2191, 1994.

[3] C. Fowlkes et al. Spectral grouping using the nystrom method. *TPAMI*, pages 214–225, 2004.

[4] Alex Gittens and Michael Mahoney. Revisiting the nystrom method for improved large-scale machine learning. *JMLR.*, 28(3):567–575, 2013.

[5] Y. Low et al. GraphLab: A new parallel framework for machine learning. *UAI*, pages 340–349, 2010.

[6] A. Mirhoseini, E. Dyer, E. Songhori, E. Baraniuk, and F. Koushanfar. Rankmap: A platform-aware framework for distributed learning from dense datasets. In *arXiv:1503.08169*, 2015.