Scalable Fast Rank-1 Dictionary Learning for fMRI Big Data Analysis

Xiang Li[1*], Milad Makkie[1*], Binbin Lin[2], Mojtaba Sedigh Fazli[1], Ian Davidson[3]
Jieping Ye[2#], Tianming Liu[1#], Shannon Quinn[1#]

[1]Department of Computer Science, University of Georgia, Athens, GA
[2]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI
[3]Department of Computer Science, University of California, Davis, CA
[*]Equal contributions; [#]Joint corresponding authors.

ABSTRACT

It has been shown from various functional neuroimaging studies that sparsity-regularized dictionary learning could achieve superior performance in decomposing comprehensive and neuroscientifically meaningful functional networks from massive fMRI signals. However, the computational cost for solving the dictionary learning problem has been known to be very demanding, especially when dealing with large-scale data sets. Thus in this work, we propose a novel distributed rank-1 dictionary learning (D-r1DL) model and apply it for fMRI big data analysis. The model estimates one rank-1 basis vector with sparsity constraint on its loading coefficient from the input data at each learning step through alternating least squares updates. By iteratively learning the rank-1 basis and deflating the input data at each step, the model is then capable of decomposing the whole set of functional networks. We implement and parallelize the rank-1 dictionary learning algorithm using Spark engine and deployed the resilient distributed dataset (RDDs) abstracts for the data distribution and operations. Experimental results from applying the model on the Human Connectome Project (HCP) data show that the proposed D-r1DL model is efficient and scalable towards fMRI big data analytics, thus enabling data-driven neuroscientific discovery from massive fMRI big data in the future.

Categories and Subject Descriptors

• **Information systems~Data mining** • Computing methodologies~Machine learning

General Terms

Algorithms, Performances

Keywords

fMRI; Sparse coding; Distributed computation; Algorithm parallelization

KDD'16, August 13–17, 2016, San Francisco, CA, USA.

1. INTRODUCTION

In recent years, the field of neuroimaging studies based on functional magnetic resonance imaging (fMRI) has featured unprecedented large-scale data availability thanks to the efforts from a series of data collection works including Human Connectome Project [26], 1000 Functional Connectomes [19] and OpenfMRI Project [20]. The rapidly growing data characterized different aspects of brain cognitive processes as well as various disorders, thus providing a great opportunity for decoding and identification of potential functional biomarkers for brain diseases. Consequently, there is an urgent call for more efficient and scalable data analytics and knowledge discovery methods, especially for dealing with fMRI big data. Functional network analysis has become an important and popular approach for discovering the underlying organization structures and meaningful dynamic patterns from the vast and noisy functional brain signals. Focusing on understanding the functional aggregation/co-activation among brain regions through quantitative and data-driven approaches, researchers have employed various types of matrix decomposition methods for the functional network analysis studies, including Independent Component Analysis (ICA) [24], Principal Component Analysis (PCA) [23] and Dictionary Learning [11]. Dictionary learning in particular has been shown to be a powerful tool in image compressed sensing [21], classification [14] and denoising [8], and has shown superior performance in decomposing the meaningful and comprehensive functional networks from various types of fMRI signals [15]. However, the computational cost for solving the sparse coding problem for dictionary learning has been known to be very demanding [12], especially when dealing with large-scale data sets. Furthermore, most of current dictionary learning methods for fMRI data analysis are only implemented for local application without any parallelization scheme. Facing with the rapidly growing fMRI data and the needs for population-level analysis with terabytes or even petabytes of data size [23], the computation power limit of a single machine will eventually become the bottleneck for efficient and effective knowledge discovery from the fMRI big data.

Following the previous success in using dictionary learning for functional network decomposition [15], in this work we devolved a novel distributed rank-1 dictionary learning (D-r1DL) model, leveraging the power of distributed computing for handling large-scale fMRI big data. Compared to the gradient-based dictionary learning algorithms such as the online dictionary learning [18] (based on stochastic gradient descent) and the K-SVD [4] (based

on gradient descent), the proposed rank-1 dictionary learning algorithm has a few critical advantages: 1) The learning process is a fix-point algorithm by alternating least squares updates, thus avoiding the tuning of the learning rate/step size while also avoiding the slow convergence near the solution; 2) The memory cost of the proposed algorithm is very low because it needs not to maintain the potentially large gradient matrix in the memory. The intermediate results will be discarded after each rank-1 basis is learned and stored, which further reduce the memory cost. More importantly, the rank-1 dictionary learning algorithm is very light-weighted regarding to the operational complexities: besides the input data, most of the routines in the algorithm will only take one vector as input and one vector as output. This feature helps the r1DL algorithm to be easily parallelized to its distributed version.

For the algorithm parallelization, in this work we used the Spark engine [3] to implement the D-r1DL algorithm. Spark is a high-performance distributed compute engine for large-scale data processing. It is similar to MapReduce, but has several distinct advantages that make it ideal for the deployment of large-scale analytics frameworks. First, its basic abstraction for distributed data, the resilient distributed dataset (RDD) [27], combines robust fault-tolerance with highly efficient data layout strategies. RDDs track their computation lineage as a directed acyclic graph; therefore, if a segment is lost, it can be easily recomputed from the lineage. These lineages can be optimized on-the-fly to minimize the overhead of the prescribed computations. Second, all operations in Spark are performed in-memory, thus significantly improving throughput of data pipelines. This is a departure from Hadoop MapReduce, in which data are serialized to disk in between map and reduce steps. Third, the Spark compute engine is much more generalizable than MapReduce, and can efficiently support highly diverse workloads. While Spark supports the map and reduce primitives from Hadoop MapReduce, it also supports graph processing [10] and streaming [28] APIs on the same compute engine, in addition to numerous functional primitives beyond *map* and *reduce*. This structural flexibility is crucial to the efficient implementation of a wide variety of distributed algorithms.

An illustration for the operational and algorithmic pipeline consisting of three layers of model specification is shown in Fig. 1. The first and foremost deliverable of this work is to provide an integrated solution for the large-scale fMRI big data analysis. Therefore, we initially deployed the proposed D-r1DL model on our in-house server (termed "in-house solution") with an integrated neuroinformatics system [17]. The neuroinformatics system provides a web-based user interface for fMRI data uploading, hosting and result post-processing [17], as illustrated in Fig. 1(a). Alternatively, we also tested deploying the D-r1DL model on the cloud computing service provided by Amazon Web Services Elastic Compute Cloud (AWS-EC2) [1], which has been widely applied for biomedical imaging researches due to its resource flexibility and ease of use. For the "AWS-EC2 solution", the data preprocessing was performed before running the D-r1DL model on it. Subroutines of the r1DL algorithm and its logic flow are illustrated in Fig. 1(b). The parallelization subroutines and its relationship with the r1DL algorithm are illustrated in Fig. 1(c).

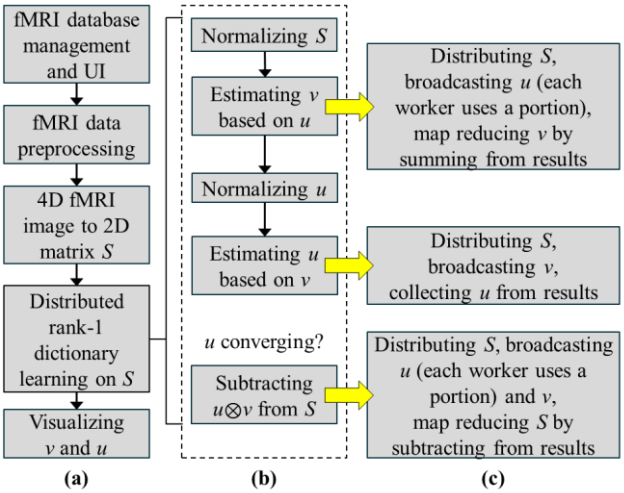

Figure 1. (a): Operations on the neuroinformatics system for preparing the application of D-r1DL and post-analysis. (b): Algorithm pipeline of the rank-1 dictionary learning. (c): Parallelization subroutines of the D-r1DL model derived from the corresponding subroutines of the r1DL using Spark. The distribution of input data *S* is based on RDDs.

2. BACKGROUND AND RELATED WORK

Functional network analysis based on matrix decomposition methods has the basic premise: the observed functional signals are the result of the linear combination from the signals of many latent source (i.e. functional networks), plus noises and/or artifacts signals [5]. The methods then aim to identify the latent source signals as well as the loading matrix based on various learning priors including spatial/temporal independence [5; 24] (for ICA), or the sparsity in the loading coefficients [11; 15] (for dictionary learning). The decomposition results consist of two parts: the signals of the latent sources (i.e. temporal pattern of the functional networks) that are regarded as basis activation patterns, and the loading matrix characterizing how each source contributes to the formation of each observed signal across voxels/ brain regions (i.e. spatial pattern of the functional networks). The spatial and temporal pattern of a sample network decomposed result is shown in Fig. 2.

Several sets of consistent and meaningful functional networks have been identified in the previous literature: including the 10 well-established resting-state networks (RSNs) [24] obtained using ICA, the task-related patterns obtained using PCA [25], and the HAFNI (holistic atlases of functional networks and interactions) atlases [15] featuring 32 group-wise consistent patterns during both task and resting-state using dictionary learning [2]. Visualization of one functional network from the matrix decomposition results using the proposed r1DL method is shown below, to illustrate the temporal (the time series curve) and spatial (on cortical volumetric space) patterns of the brain functional networks.

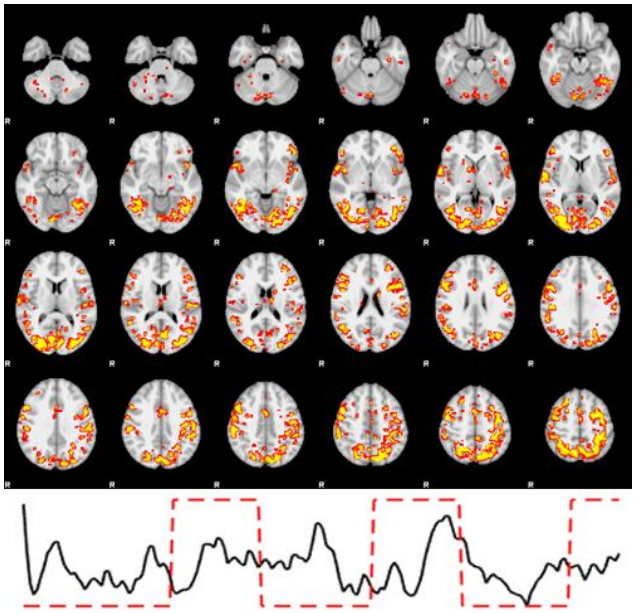

Figure 2. Top: spatial pattern visualized on cortical volumetric space of one decomposed network. Bottom: visualization of its temporal pattern.

While dictionary learning in general is an active area of research, there has been significantly less effort in scaling the algorithm. One of the few methods proposed was [22], in which the authors designed a sparse coding framework on Hadoop MapReduce. The method was parallelized by splitting the core operations in two main phases: the sparse coding phase, in which the loading weights were learned in parallel; and the dictionary learning phase, in which the dictionary atoms were updated. By taking advantage of hard sparsity constraints, the authors avoided materializing the entire data matrix in memory at once, instead operating on blocks of the matrix in parallel and constructing the loading matrix row by row. In this work we undertook the similar dataflow optimization techniques: the sparse coding and dictionary atoms are assumed to fit easily in memory for fast and efficient computation. However, using the Spark framework instead of Hadoop MapReduce provides us intrinsic speedups. Where Hadoop MapReduce excels in batch processing, Spark is optimized for iterative computation: intermediate results are cached in-memory on the workers and re-used in subsequent iterations, and the updates are efficiently broadcasted to the workers. Most importantly, the RDDs abstraction pipelines the requested operations and lazily executes them after determining the optimal computational path using the least amount of resources. We leverage these advantages to provide a substantial performance gain in our D-r1DL dictionary learning implementation.

3. RANK-1 DICTIONARY LEARNING FOR FMRI DATA ANALYSIS

The rank-1 dictionary learning algorithm aims to iteratively estimate multiple rank-1 basis vector u (T×1 vector with unit length) and its loading coefficient vector v (P×1 vector) to decompose the input signal matrix S of dimension T×P, by minimizing the following energy function $L(u, v)$:

$$L(u,v) = \|S - uv^T\|_F, \text{s.t.}\ \|u\| = 1, \|v\|_0 \le r. \qquad (1)$$

Eq. 1 indicates that the product of u and v is supposed to well-fit the input S while the total number of non-zero elements in v should be smaller than or equal to the given sparsity constraint parameter r. The minimization problem in Eq. 1 can be solved by alternatively updating u (randomly initialized before the first iteration) and v until convergence:

$$v = \underset{v}{\text{argmin}}\|S - uv^T\|_F, s.t. \|v\|_0 \le r,$$

$$u = \underset{u}{\text{argmin}}\|S - uv^T\|_F = \frac{Sv}{\|Sv\|}, \qquad (2)$$

Converging at step j if: $\|u^{j+1} - u^j\| < \varepsilon, \varepsilon = 0.01$.

Eq. 2 involves multiplication between input matrix S and vector u, followed by setting all elements in the resulting vector smaller than its r-th largest value to zero, essentially performing the vector partition operation. One rank-1 basis [u, v] can be estimated in each step; afterwards the input matrix S will be deflated to its residual R:

$$R^n = R^{n-1} - v^T R^{n-1}, R^0 = S, 1 < n \le K, \qquad (3)$$

where K is the total number of expected basis (i.e. dictionary atoms) to be discovered from the input data. It could be seen that the formulation of the proposed rank-1 dictionary learning algorithm is similar to the sparse PCA problem [6]. However, the goal of PCA and sparse PCA is to derive a low-dimensional basis (i.e. learning a smaller set of high representative basis); in contrast, the goal in dictionary learning is to learn an over-complete dictionary set [12]. Regarding the algorithm convergence, it is easy to show that the value of the energy function as in Eq. 1 decreases at each iteration (until convergence), thus the objective is guaranteed to converge. The convergence of the learning in Eq. 2 was also empirically tested in this work. The deflation operation in Eq. 3 is based on Hotelling's deflation method for estimating the eigenvectors, where each step of deflation leads to the corresponding eigenvalue replaced by zero. The validity of using Hotelling's deflation for sparse PCA was provided in [6], while better deflation methods were also discussed in [16]. Fig. 3 shows a running example illustrating the data preparation and networks decomposed by r1DL.

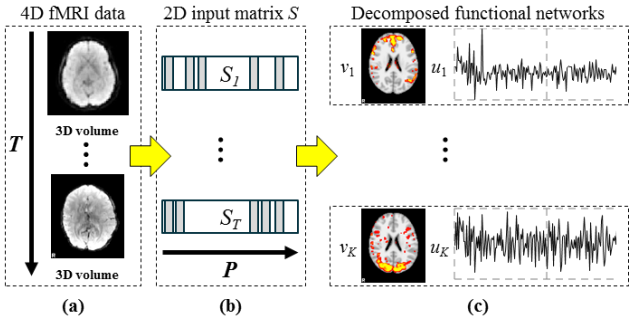

Figure 3. Illustration of the r1DL model applied on fMRI data as a running example. (a) 4D fMRI data represented as a series of 3D volume images. (b) Converting 4D data into 2D input matrix *S*. (c) Spatial (v_1…v_K) and temporal (u_1…u_K) patterns of the decomposed functional networks from *S* using r1DL dictionary learning.

4. ALGORITHM PARALLLIZTAION AND DEPLOYMENT ON SPARK

In this work, the rank-1 dictionary learning algorithm introduced above was implemented and parallelized on the Spark engine. Specifically, the vector-matrix multiplication and the matrix-vector multiplication steps in Eq. 2 were implemented by their corresponding distributed primitives in Spark. Reading and partitioning the input data S is supported by the RDD abstraction; therefore, the distribution of S to each node as a series of key-value pairs is inherently straight forward: data formation of the current work is based on row-vectors. In other words, each column in S contains the T number of observations for one specific feature, to the total of P features. While S was maintained as an RDD, the vectors u and v were broadcast to all nodes. Thus during the vector-matrix multiplication, each node will use its portion of the updated u vector, and then estimate the v vector based on the multiplication of portions of S and u. The resulting v vectors from all the nodes will be then map-reduced by the summation operation. The matrix-vector multiplication is relatively easier, where each node will use all the updated v vector then estimate its corresponding portion of the u vector. The resulting u vector is just the collection of the results from each node. In addition, the matrix deflation operation in Eq. 3 was also parallelized by broadcasting both the u and v vectors learned from Eq. 2, and then estimating the outer produce between portion of u vector and the whole v vector at each node. The S matrix is then subtracted by the results of each node through mapping over each row and deflating it in parallel. This implementation of the r1DL algorithm after parallelization is termed as the "Distributed rank-1 dictionary learning" (D-r1DL) model.

4.1 Complexity of the distributed primitives

The computational complexity of the original (un-parallelized) rank-1 dictionary learning algorithm is quite obvious: all the major subroutines including matrix-vector multiplication, vector-matrix multiplications and deflation have complexity of $T*P$, essentially traversing through input matrix S. However, the distributed primitives added for the parallelization in D-r1DL will potentially cause large extra computations and/or data transfers across nodes. The problem will be magnified when such transfer occurs over a network (e.g. worker nodes are distributed across Internet). Thus we analyzed the extra complexity induced by the parallelization of the three subroutines, assuming that there are M number of nodes. It should be noted that the estimations are upper bounds for the complexity; the empirical performance will be largely dependent on how Spark optimizes the transformation lineage for the RDD, and how the RDD is distributed across the cluster.

- For the vector-matrix multiplication for estimating v, the total complexity is ($T*M + P*M+T\log(T) + P$): $T*M$ caused by the broadcasting, $P*M+T\log(T)$ caused by the map reduce and network shuffle, and P caused by the updating of v.
- For the matrix-vector multiplication, the total complexity is ($P*M + T$): $P*M$ caused by the broadcasting, and T caused by the updating of u.
- For the matrix deflation, the total complexity is ($P*M+T*M$): both u and v will be broadcasted to all M nodes.

4.2 Deployment and configuration

The D-r1DL model was deployed on two different sets of server clusters, leading to two solutions for the data analysis. The first set is based on the in-house server, called the "in-house solution". Spark version 1.5.2 pre-built for Hadoop 1.X and python version 2.7.11 with all the required dependencies were installed on the in-house server. We setup one standalone spark cluster on the server with a master node consisting of 16 cores and 16GB RAM. The in-house solution also featured an integrated neuroinformatics system named HELPNI (HAFNI-enabled largescale platform for neuroimaging informatics) as introduced in [17]. Authorized users of the system can upload, manage, and perform the preparations of the fMRI data for the model analysis. For running the D-r1DL model, the preparation steps include fMRI signal preprocessing (gradient distortion correction, motion correction, bias field reduction, and high pass filtering) [9], converting the 4D fMRI images to 2D data matrix, as well as the generation of shell scripts according to user specifications. When the computations by D-r1DL are finished, the reports of the results (consisting of the statistics and visualizations of the decomposed functional networks) will be automatically generated by the HELPNI neuroinformatics system [17] and uploaded to the server, accessible via web interface for viewing and sharing. The in-house solution is mainly used for testing the single-server (on the local threads), standalone mode performance of the D-r1DL algorithm.

The second set was based on the computational resources provided by AWS-EC2 service. For the AWS-EC2 implementations, we used the provided EC2 deployment scripts with the Spark distribution. These created a Spark cluster on EC2 with one master node and 16 workers. Each worker consisted of 2 cores and 7.5GB memory. EC2 clusters are highly scalable, as the number of workers recruited could be adjusted within the cluster. The preprocessed and converted fMRI data was stored at the cloud through Amazon S3 and accessible by the EC2 cluster. The nodes featured Hadoop distributed file system (HDFS), which ensured data consistency and improved I/O speed. Hadoop version 1.0, Spark version 1.5.2, and Python version 2.7.11 with the Anaconda scientific programming stack (e.g. NumPy, SciPy) were installed on EC2 cluster. An illustrative diagram showing the organization and execution architecture of the two solutions are shown below.

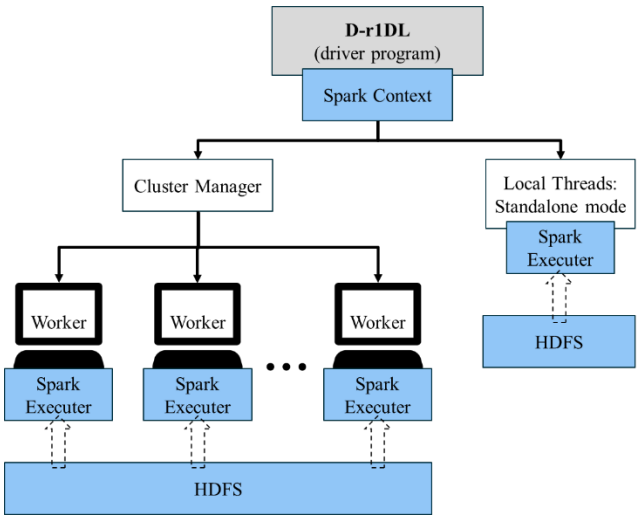

Figure 4. Illustrative diagram showing the organization and execution architectures for the standalone local mode and the multiworker cluster mode.

5. EXPERIMENTAL APPLICATION ON BRAIN FUNCTIONAL IMAGING DATA

5.1 Validation of the D-r1DL model

To validate the effectiveness of the proposed D-r1DL model in terms of its capability of decomposing fMRI data into meaningful functional networks, we applied the framework on the fMRI data acquired during multiple tasks from the Human Connectome Project (HCP) Q1 release dataset [26]. The HCP dataset is advantageous in its high temporal and spatial resolution (TR=0.72s, varied temporal length from 176 to 1200 volumes; 2mm isotropic voxels, to the total of over 200,000 voxels), which enables more detailed characterization of the brain's functional behavior. Additionally, the HCP dataset includes acquisitions of fMRI data during 7 tasks and resting-state from 68 subjects, to the total of over 500 individual data, which matches the aim of the proposed framework for population-level fMRI big data analysis.

The learned collection of functional networks represented by rank-1 dictionary basis was then compared with the HAFNI atlases [15]. The HAFNI atlas was obtained by applying online dictionary learning method [18] on all the individual fMRI data in the same HCP Q1 dataset. Subsequently, 32 group-wise consistent networks were identified through manual inspection over more than 200,000 decomposed networks [15]. Thus our aim in this validation was to identify the presence (or absence) of those atlas networks from the results of D-r1DL. The main rationale of performing comparison with network templates and atlas network for the model validation in this work was due to the lack of ground truth in fMRI data. At this stage, the capability and accuracy of the proposed model could only be validated by comparing with the previously-reported and well-established results from the same dataset.

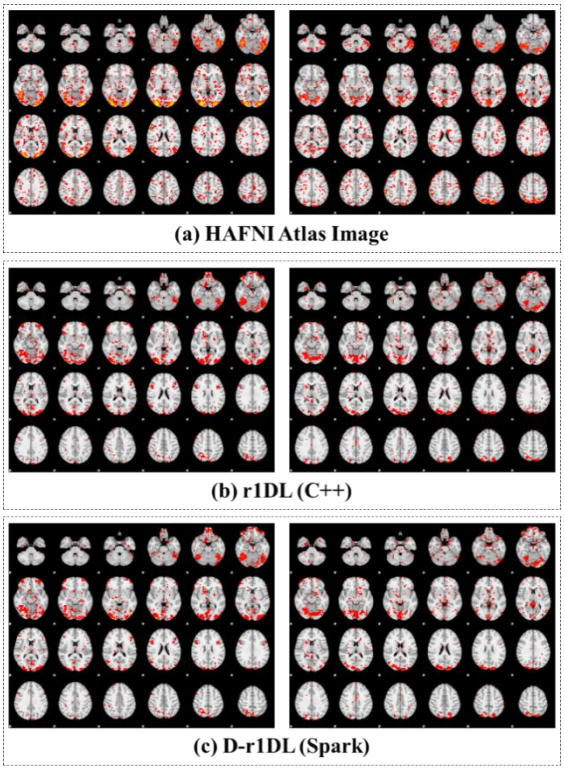

Figure 5. Spatial maps obtained from applying the dictionary learning method on the same fMRI dataset implemented by SPAMS, r1DL in C++, and D-r1DL in Spark.

In addition, the D-r1DL model was compared with the rank-1 dictionary learning algorithm implemented in C++ without any parallelization, in order to investigate whether the parallelization would affect the model performance. Both of the implementations were deployed on the same in-house server and applied on the HCP Q1 individual resting-state fMRI data as well as two task fMRI datasets ("Emotion" and "Working Memory") using the same parameter setting (r=0.07, K=80). An illustration for the spatial maps of two sample networks decomposed by r1DL implemented in C++ and D-r1DL, as well as the corresponding individual-level atlas network in HAFNI, is shown in the three panels in Fig. 5 (a)-(c), respectively.

The obtained functional networks from D-r1DL were then matched to the individual-level atlas networks on the same subjects of the same tasks (or resting-state) by maximizing the spatial similarity between them:

$$R(P_1, P_2) = |P_1 \cap P_2| / |P_2|, \quad (4)$$

where P_1 and P_2 are the spatial map vectors of the two networks. In this work P_1 is the network decomposed by D-r1DL and P_2 is the atlas network. Operator $|\bullet|$ counts the total number of voxels with non-zero values in the given spatial pattern. R ranges from 0 (no voxels overlapping) to 1 (exactly the same maps). The spatial similarity results show that all of the atlas networks defined in HAFNI could be found from the results of D-r1DL. Specifically, the 3 atlas networks from Emotion dataset were identified from D-r1DL results with average spatial similarity of 0.82. The 6 atlas networks from WM dataset were identified with average spatial similarity of 0.79. The 10 resting-state networks, originally reported in [7] and later included in the HAFNI atlas were identified from the results of applying D-r1DL on resting state fMRI data with average spatial similarity of 0.70. The spatial similarity results also show that the networks decomposed by r1DL implemented in C++ and D-r1DL implemented in Spark are almost identical, with average spatial similarity of 0.99.

5.2 Experiment on algorithm convergence

As discussed in section 3 for the algorithm description, we have performed extensive experiments for testing the convergence of alternating least squares with l-0 constraint problem in Eq. 2. By running the r1DL algorithm on each individual fMRI dataset in the HCP Q1 release across 7 tasks with dictionary size parameter K=400 (i.e. decomposing 400 functional networks from each fMRI data), we found that the learning of u and v converged for all the 190,400 networks decomposed. The average number of alternative updates needed for learning one network for different datasets of 5 randomly selected sample subjects is listed in Table 1 below. It can be seen that regardless of the input data size, the majority of the learning would be finished within only a few iterations.

Table 1. Average number of iterations needed for convergence across 7 tasks of 5 subjects

	sbj1	**sbj2**	**sbj3**	**sbj4**	**sbj5**
Emotion	6.1	6.1	6.2	6.1	6.2
Gambling	6.3	6.3	6.5	6.3	6.2
Language	6.4	6.3	6.4	6.3	6.3
Motor	6.4	6.2	6.3	6.5	6.3
Relational	6.2	6.2	6.2	6.2	6.2
Social	6.3	6.3	6.2	6.3	6.3
WM	6.4	6.3	6.4	6.4	6.4

5.3 Performance boost by r1DL comparing with other dictionary learning algorithms

One of a major premise of the proposed r1DL algorithm is that because of its smaller memory cost and robust learning mechanism (no need to set learning rate), the algorithm should have similar or faster running speed, compared with other dictionary learning methods, even without the parallelization. Based on the performance statistics from running r1DL over the whole HCP task fMRI (tfMRI) dataset as introduced above, we compare the r1DL with the other two dictionary learning algorithms: online dictionary learning implemented in SPAMS package [18] and the stochastic coordinate coding (SCC) algorithm introduced in [13], by applying these two methods on the same HCP Q1 dataset using the same in-house server. The performance comparison is shown in Fig. 6 (averaged across all 68 subjects). From the comparison, it can be seen that r1DL has exhibited improved computational speed over the other two methods in all the 7 tfMRI datasets. It should be noted that we used the r1DL implemented in C++ for the testing in this experiment, and in the same way the other two methods were implemented to ensure consistency in the comparison.

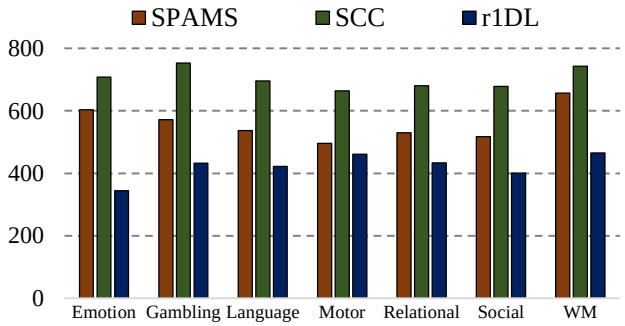

Figure 6. Average time cost (measured in seconds) for functional network decomposition from individual tfMRI data during 7 tasks across 68 subjects, using the three dictionary learning methods.

5.4 Performance and scalability analysis for D-r1DL using the in-house solution

As introduced in 5.1, in this work we applied the D-r1DL model on the three types of datasets for functional network decompositions: tfMRI data of Emotion task with dimension of 176*2M, tfMRI data of Working Memory (WM) task with dimension of 405*2M, and resting state fMRI (rsfMRI) data with dimension of 1200*2M. The testing input files sizes of the three types of dataset were 300MB, 700MB and 2GB, respectively. Using the in-house solution as specified in 4.2, we firstly analyzed the performance of the D-r1DL model using different numbers of cores on a single machine. The performance statistics measured in time cost are shown in Fig. 7.

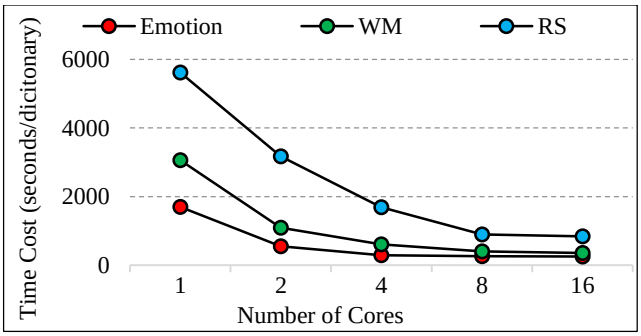

Figure 7. Time cost for decomposing one functional network from three different fMRI datasets by recruiting varying number of cores, using the in-house solution.

For all the three datasets, there exists clear logarithmic relationship (R^2=0.84, 0.89 and 0.92) between the number of cores recruited and the total time cost for the decomposition. The speed boosts by recruiting more cores for the computation comparing with the baseline (1-core) configuration for the three datasets using the in-house solution are listed in Table 2, showing the ratio between the time cost using 1 core and the time cost using multiple cores.

Table 2. Ratios of time cost decreases by recruiting more cores comparing with the single-core configuration

	Emotion	**WM**	**RS**
2 cores	3.1	2.8	1.8
4 cores	6.0	5.1	3.3
8 cores	6.6	7.7	6.3
16 cores	6.8	8.6	6.7

As the configuration for using only one core for D-r1DL is equivalent to the non-parallel algorithm, the performance statists indicate that the parallelization based on Spark could greatly improve the performance of the rank-1 dictionary learning algorithm. Specifically, the better performance gain on larger dataset indicates that the parallelization of the rank-1 dictionary learning could potentially overcome the computational bottleneck for analyzing big neuroimaging data, potentially enabling high-throughput analysis on a locally-deployed high-performance computation cluster in the future.

Another analysis of the performance of D-r1DL on the in-house server was aiming at investigating the relationship between dictionary size K (i.e. number of functional networks to be decomposed) and the time/memory cost. The rationale is that, as discussed in 3.1, the rank-1 dictionary learning algorithm has advantages in the iterative estimation of the basis vectors [u, v]. Thus the program does not need to maintain the learning results in the memory. As shown in Fig. 8, the average time cost for estimating one dictionary only marginally increased when using larger dictionary sizes K. Further, the total memory cost of running the D-r1DL was independent with K. This feature is especially useful when the spatial dimension of the input data is large, either because of the higher spatial resolution or because of the aggregated dataset from multiple subjects in a population. As in such cases, the size of all decomposed networks, which equals to $P*K$, could be very large.

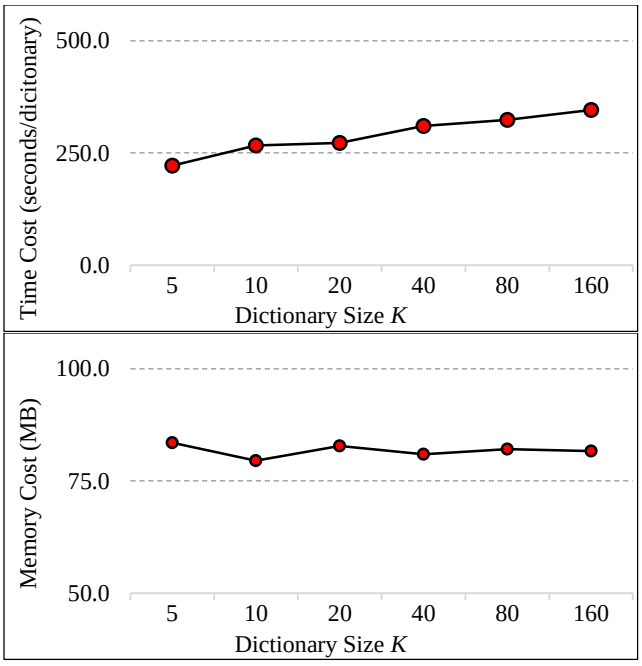

Figure 8. Time and memory costs for decomposing Emotion tfMRI datasets with varying dictionary size *K*, recruiting 16 cores, using the in-house solution.

5.5 Performance of D-r1DL in multi-worker mode using AWS-EC2 solution

In addition to the experiments of the single-machine multi-core configurations conducted using the in-house solution, we have also applied the D-r1DL on the same datasets using the cloud computing service provided by AWS-EC2 as introduced in 4.2. We aimed to investigate the performance of D-r1DL when applied over multiple machines through a network interface. Specifically, as the Spark Python architecture and the resilient distributed dataset abstracts have been designed for supporting large-scale, high efficient analytic framework, we are interested to test its capability of utilizing the distributed computational resources provided by AWS-EC2. In this work, we tested the performance in terms of time and memory cost of the D-r1DL model using 1, 2, 4, 8 and 16 workers on three datasets, while each worker has two cores for the computation. It should be noted that for normal usage, 16 is the maximum number of workers allowed for one instance for the EC2 configuration. The D-r1DL would be running in stand-alone mode under single-worker configuration, similar to the configuration used in the in-house solution. As discussed in 4.1, the communications through network interfaces caused by the parallelization of computation (e.g. the broadcasting of u and v) will potentially increase the time cost mainly due to latencies. Thus the single-worker configuration serves as the baseline for testing whether recruiting more workers will be beneficial from the performance perspective. The performance results of the time and memory cost are summarized in Fig. 9, with the baseline results from single-worker configuration highlighted.

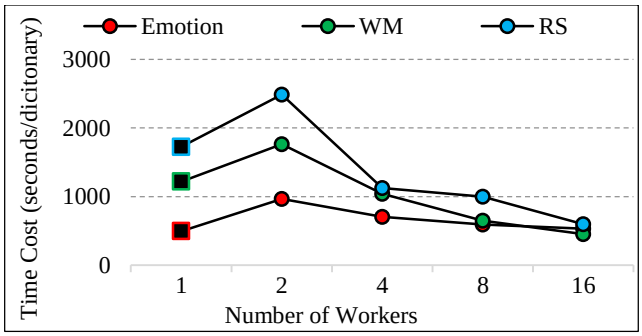

Figure 9. Time cost for decomposing one functional network from three different fMRI datasets by recruiting varying numbers of workers, using the AWS-EC2 solution. The results from single-worker (standalone mode) configuration are highlighted as blocked markers.

First of all, from the results it can be seen that the AWS-EC2 solution recorded faster computation speed (10%~80% faster) comparing with the in-house solution, especially on larger dataset, when both of them use two cores. Considering the fact that the hardware configuration of AWS-EC2 features larger memory capacity better optimized for computation purposes, such difference in performance is within our expectation.

On the other hand, it is interesting to observe that for AWS-EC2 solution, there exists the break-even point at which the multiple-worker mode outperformed the stand-alone mode, but only for the two larger datasets. The ratio between the time cost using standalone mode and the time cost using multiple workers are for the three datasets summarized in Table 3. It can be observed that, for the 700MB WM and the 2GB RS dataset, using 4 or more workers could lead to faster speed comparing with the standalone mode using 1 worker. While for the smaller 300MB Emotion data, the standalone mode is the fastest among all experiments. Thus it can be concluded that the multi-worker configuration will be more suitable for analyzing larger datasets, while standalone mode or the simpler in-house server solution might be preferred for datasets with typically smaller sizes.

Table 3. Ratios of time cost changes by recruiting various number of workers comparing with standalone mode

	Emotion	**WM**	**RS**
2 workers	3.1	2.8	1.8
4 workers	6.0	5.1	3.3
8 workers	6.6	7.7	6.3
16 workers	6.8	8.6	6.7

The memory cost on each worker as summarized in Fig. 10 indicates that the multi-worker mode under AWS-EC2 solution scales good with the increasing input file size, as it maintains reasonable small (~100MB) memory cost for all configurations including the single-worker standalone mode. That is the major advantage of using Spark Python model and its resilient distributed dataset for the parallelization: one or multiple workers need not to load the whole dataset at once, but only its corresponding portion of the data according to the data partitioning strategy implemented in the RDDs abstract.

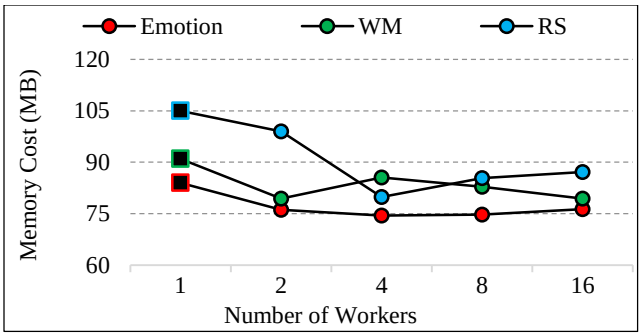

Figure 10. Memory cost for decomposing three different fMRI datasets by recruiting varying number of workers, using the AWS-EC2 solution. The results from single-worker (standalone mode) configuration are highlighted as blocked markers.

6. CONCLUSIONS

In this paper, we proposed a novel and effective distributed dictionary learning model based on iterative rank-1 basis estimation. The model was implemented and parallelized in Spark, and then deployed using the in-house solution as well as the AWS-EC2 solution. The aim of this work is to meet the challenges posed by fMRI big data for more efficient and scalable data analytics methods. The testing results from running both solutions on the HCP Q1 dataset show that functional network decomposition using rank-1 dictionary learning could benefit from parallelization for both single-worker multi-core configuration and the multi-worker cluster configuration, with significant performance improvement especially on larger datasets. In the current work, we have only analyzed individual-level fMRI data, with the largest data size of 2GB. As it has been shown from the performance statistics that the Spark engine supported by RDDs abstract could effectively perform the data partition and reduce the memory cost for large-scale input data, we will test the model performance on larger, population-level datasets (e.g., the HCP full dataset) with the size of dozens or hundreds of terabytes in the near future. The ultimate goal of the proposed D-r1DL model with the HELPNI neuroinformatics system is to provide an integrated solution for functional neuroimaging big data management and analysis, enabling high-throughput neuroscientific knowledge discovery. In addition, the similar parallelization scheme used in this work for D-r1DL could be implemented on other algorithms as well. Thus the experience of this work also offers a practical perspective for improving the efficiency and scalability of general machine learning and data mining algorithm developments.

7. REFERENCES

[1] https://aws.amazon.com/ec2/

[2] http://hafni.cs.uga.edu

[3] https://spark.apache.org

[4] Aharon, M., Elad, M., and Bruckstein, A., 2006. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *Signal Processing, IEEE Transactions on 54*, 11, 4311-4322. DOI= http://dx.doi.org/10.1109/TSP.2006.881199.

[5] Biswal, B.B. and Ulmer, J.L., 1999. Blind Source Separation of Multiple Signal Sources of fMRI Data Sets Using Independent Component Analysis. *Journal of Computer Assisted Tomography 23*, 2, 265-271.

[6] D'aspremont, A., Ghaoui, L.E., Jordan, M.I., and Lanckreit, G.R., 2004. A Direct Formulation for Sparse PCA Using Semidefinite Programming. In *Advances in Neural Information Processing Systems*.

[7] Damoiseaux, J.S., Rombouts, S.A.R.B., Barkhof, F., Scheltens, P., Stam, C.J., Smith, S.M., and Beckmann, C.F., 2006. Consistent resting-state networks across healthy subjects. *Proceedings of the National Academy of Sciences of the United States of America 103*, 37, 02/20/received), 13848-13853. DOI= http://dx.doi.org/10.1073/pnas.0601417103.

[8] Elad, M. and Aharon, M., 2006. Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *Image Processing, IEEE Transactions on 15*, 12, 3736-3745. DOI= http://dx.doi.org/10.1109/TIP.2006.881969.

[9] Glasser, M.F., Sotiropoulos, S.N., Wilson, J.A., Coalson, T.S., Fischl, B., Anderson, J.L., Xu, J., Jbabdi, S., Webster, M., Polimeni, J.R., Van Essen, D.C., and Jenkinson, M., 2013. The minimal preprocessing pipelines for the Human Connectome Project. *NeuroImage 80*, 105-124. DOI= http://dx.doi.org/http://dx.doi.org/10.1016/j.neuroimage.2013.04.127.

[10] Gonzalez, J.E., Xin, R.S., Dave A., Crankshaw, D., Franklin, M.J., and Stoica, I., 2014. GraphX: graph processing in a distributed dataflow framework. In *Proceedings of the Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation*, USENIX Association, 2685096, 599-613.

[11] Kangjoo, L., Sungho, T., and Jong Chul, Y., 2011. A Data-Driven Sparse GLM for fMRI Analysis Using Sparse Dictionary Learning With MDL Criterion. *Medical Imaging, IEEE Transactions on 30*, 5, 1076-1089. DOI= http://dx.doi.org/10.1109/TMI.2010.2097275.

[12] Lee, H., Battle, A., Raina, R., and NG, A.Y., 2006. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*.

[13] Lin, B., Li, Q., Sun, Q., Lai, M.-J., Davidson, I., Fan, W., and Ye, J., 2014. Stochastic Coordinate Coding and Its Application for Drosophila Gene Expression Pattern Annotation. *arXiv:1407.8147*.

[14] Liu, B.-D., Wang, Y.-X., Zhang, Y.-J., and Shen, B., 2013. Learning dictionary on manifolds for image classification. *Pattern Recognition 46*, 7, 1879-1890. DOI= http://dx.doi.org/http://dx.doi.org/10.1016/j.patcog.2012.11.018.

[15] Lv, J., Jiang, X., Li, X., Zhu, D., Zhang, S., Zhao, S., Chen, H., Zhang, T., Hu, X., Han, J., Ye, J., Guo, L., and Liu, T., 2015. Holistic atlases of functional networks and interactions reveal reciprocal organizational architecture of cortical function. *Biomedical Engineering, IEEE Transactions on 62*, 4, 1120-1131. DOI= http://dx.doi.org/10.1109/TBME.2014.2369495.

[16] Mackey, L.W., 2008. Deflation Methods for Sparse PCA. In *Advances in Neural Information Processing Systems*.

[17] Makkie, M., Zhao, S., Jiang , X., Lv , J., Zhao, Y., Ge, B., Li, X., Han, J., and Liu, T., 2015. HAFNI-enabled largescale platform for neuroimaging informatics (HELPNI). Brain Informatics 2, 4, 225-238. DOI= http://dx.doi.org/10.1007/s40708-015-0024-0.

[18] Mairal, J., Bach, F., Ponce, J., and Sapiro, G., 2010. Online Learning for Matrix Factorization and Sparse Coding. *J. Mach. Learn. Res. 11*, 19-60.

[19] Mennes, M., Biswal, B.B., Castellanos, F.X., and Milham, M.P., 2013. Making data sharing work: The FCP/INDI experience. *NeuroImage 82*, 683-691. DOI= http://dx.doi.org/http://dx.doi.org/10.1016/j.neuroimage.2012.10.064.

[20] Poldrack, R.A., Barch, D.M., Mitchell, J.P., Wager, T.D., Wagner, A.D., Devlin, J.T., Cumba, C., Koyejo, O., and Milham, M.P., 2013. Toward open sharing of task-based fMRI data: the OpenfMRI project. *Frontiers in Neuroinformatics 7*, 12. DOI= http://dx.doi.org/10.3389/fninf.2013.00012.

[21] Ravishankar, S. and Bresler, Y., 2011. MR Image Reconstruction From Highly Undersampled k-Space Data by Dictionary Learning. *Medical Imaging, IEEE Transactions on 30*, 5, 1028-1041. DOI= http://dx.doi.org/10.1109/TMI.2010.2090538.

[22] Sindhwani, V. and Ghoting, A., 2012. Large-scale distributed non-negative sparse coding and sparse dictionary learning. In *Proceedings of the Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining* 2339610, 489-497. DOI= http://dx.doi.org/10.1145/2339530.2339610.

[23] Smith, S.M., Hyvarinen, A., Varoquaux, G., Miller, K.L., and Beckmann, C.F., 2014. Group-PCA for very large fMRI datasets. *NeuroImage 101*, 0, 738-749. DOI= http://dx.doi.org/http://dx.doi.org/10.1016/j.neuroimage.2014.07.051.

[24] Smith, S.M., Miller, K.L., Moeller, S., XU, J., Auerbach, E.J., Woolrich, M.W., Beckmann, C.F., Jenkinson, M., Andersson, J., Glasser, M.F., Van Essen, D.C., Feinberg, D.A., Yacoub, E.S., and Ugurbil, K., 2012. Temporally-independent functional modes of spontaneous brain activity. *Proceedings of the National Academy of Sciences 109*, 8, 3131-3136. DOI= http://dx.doi.org/10.1073/pnas.1121329109.

[25] Thirion, B. and Faugeras, O., 2003. Dynamical components analysis of fMRI data through kernel PCA. *NeuroImage 20*, 1, 34-49. DOI= http://dx.doi.org/http://dx.doi.org/10.1016/S1053-8119(03)00316-1.

[26] Van Essen, D.C., Smith, S.M., Barch, D.M., Behrens, T.E.J., Yacoub, E., and Ugurbil, K., 2013. The WU-Minn Human Connectome Project: An overview. *NeuroImage 80*, 0, 62-79. DOI= http://dx.doi.org/http://dx.doi.org/10.1016/j.neuroimage.2013.05.041.

[27] Zharia, M., Chowdhury, M., Das, T., Dave, A., MA, J., Mccauley, M., Franklin, M.J., Shenker, S., and Stoica, I., 2012. Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, 2228301, 2-2.

[28] Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., and Stoica, I., 2013. Discretized streams: fault-tolerant streaming computation at scale. In *Proceedings of the Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, 2522737, 423-438. DOI= http://dx.doi.org/10.1145/2517349.2522737.