A Method to Process Images Data and Prediction Models for some MapReduce Applications

Booklet for PhD. Dissertation

Li YangYuan

Scientific Supervisor
Prof. Van Tien Do DSc
Budapest University of Technology and Economics

Budapest, Hungary
2020
1 Introduction

With the daily flourishing of digital technology and information system, big data is being explosively generated everyday. As a consequence, big data with large volume and high complexity characteristics often incur the unavailability of tradition data management and analytic tools. Reducing data dimensions prior to data analytics is a main approach to alleviate complexity problem of big data. Additionally, the distributed computing plays an important role in processing large scale dataset. With rapidly increasing of data volume, maximizing resource efficiency of such system has become more crucial than ever. Many efforts have devoted to developing dimensionality reduction algorithm and improving resource efficiency of distributed computing system.

Dimensionality reduction method plays a crucial role in alleviating the ineffectiveness of traditional statistical analysing approach in many fields [21, 29]. Particularly, most of image data has high dimensions and always confronts dimensionality curse problem. In past decades, numerous 1-dimensional dimensionality reduction methods for 2D image data have been proposed [11, 13, 20, 16, 28]. However, the processing of transforming 2-dimensional data matrices to 1-dimensional vectors may destroy the structures of data and increase the computing complication. Additionally, due to difficulty of obtaining tag information, 2D unsupervised dimensionality methods attract increasing attentions. In such dimensionality reduction methods, similarity matrix plays an important role for its efficiency and comprehensibility [1, 12, 5, 26]. Unfortunately, the existence of noises in the collected data always makes the learned similarity matrix may not be the optimal one [17, 6, 7, 19, 23, 25]. To solve the problems above, many efforts have been devoted to find a similarity matrix [17, 6, 7, 19, 23, 25]. However, most of the spectral-based clustering methods are not able to comprehensively integrate these techniques and often lead to overfitting and degrade the clustering performance [23, 25]. Therefore, an efficient dimensionality reduction algorithm for 2D unlabeled image data is greatly needed.

Hadoop is a popular distributed computing framework and widely used to big data analytics in many industrial domains. MapReduce application is an important implementation of Hadoop for big data analytics. The accurate resource usage prediction for MapReduce application makes more sense for improving resource efficiency of distributed computing system. As we know, resource usage prediction constitutes one of particularly significant tasks in the operation of computing clusters. Either cloud providers or cluster operators can manage their resource by estimating future usage requirements from the current and past usage patterns of resources. A number of researches have used multiple linear regression (MLR) to predict performance metrics of MapReduce application [15, 27, 24]. Additionally, a few researches focus on CPU usage prediction based on configuration parameters of distributed computing platform [18]. However, the constructed performance models above mainly focus on predicting execution time of Hadoop jobs and could not be used by both users/consumers and service providers in the cloud for effective resource utilization. Therefore, characterizing and establishing forecasting model for usage parameters of MapReduce application have a great need for improving resource efficiency of the specific distributed computing system.
2 Problems and Objectives

The dissertation addressed the challenging problems on the dimensionality reduction of 2D unlabeled image data, the characterizations of MapReduce applications and the prediction of the usage parameters of MapReduce applications. The problems and objectives of the dissertation are summarised in Table 1.

Thesis 1: A novel efficient dimensionality reduction algorithm for 2D unlabeled image data

| Problem | Current 2D unsupervised dimensionality reduction method is prone to incur overfitting and degrade clustering performance for 2D unlabeled image data |
| Objective | Develop an efficient dimensionality reduction method to alleviate overfitting and improve clustering performance of 2D unlabeled image data |
| Application | Extract informative features for unlabeled 2-dimensional image data |

Thesis 2: Characterization and Prediction models for MapReduce applications

| Problem | Uncertain catalog characteristics and resource dependency pattern of MapReduce application, the difficulty of accurately forecasting resource usage of MapReduce application |
| Objective | 1) Characterize MapReduce applications for categorical goal, 2) design accurate resource dependent models for several specific MapReduce applications to forecast the relationships between usage parameters, 3) establish usage parameters forecasting models for several MapReduce applications in the specific distributed computing system, extend prediction to heterogeneous machine. |
| Application | MapReduce application categorization, usage parameters prediction, optimize resource allocation strategy, optimize resource scheduling algorithm |

Table 1. Summary of research topics

3 Methodology and Research Methods

In this dissertation, the analytical and the mathematical statistics methodology are applied to study dimensionality reduction algorithm and to establish prediction models for MapReduce applications.

- To develop an efficient dimensionality reduction algorithm for 2D unlabeled image data, an operational research is applied. An optimization algorithm is developed to search the minimum of the objective function. Moreover, the convergence property of the proposed algorithm is proved.

- To characterize MapReduce applications and to build forecasting models, mathematical statistics is applied.

Applied mathematical models and algorithms are summarized in Table 2. The algorithm developed in thesis 1 are evaluated and implemented in MATLAB software, the computation, analysis, and modeling in thesis 2 are performed in R software and Python. All experimental datasets come from real-world data and real cases.
4 New Results

4.1 A novel efficient dimensionality reduction algorithm

**Thesis Group 1:** Property and performance of the “Discriminative Unsupervised 2D Dimensionality Reduction with Graph Embedding” algorithm

**4.1.1 Definition and algorithm**

Given a 2D image data set $X_1, X_2, \ldots, X_N$, where $N$ is the number of data points. Suppose these $N$ data points are sampled from $c$ clusters. For each data point $X_i \in \mathbb{R}^{m \times n}$, which can be connected by data points with the probability $P_{ij}$, and the probability represents the similarity between data points $X_i$ and $X_j$, $(i, j = 1, 2, \ldots, N)$. $U \in \mathbb{R}^{m \times u}$ and $V \in \mathbb{R}^{n \times v}$ represent the row-directional and column-directional projection matrices respectively. Notations are presented in Table 3.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>The number of data points</td>
</tr>
<tr>
<td>$c$</td>
<td>Cluster number</td>
</tr>
<tr>
<td>$L$</td>
<td>Laplacian matrix</td>
</tr>
<tr>
<td>$P$</td>
<td>Similarity matrix</td>
</tr>
<tr>
<td>$P_i$</td>
<td>The $i$-th column of matrix $P$, $(i = 1, 2, \ldots, N)$</td>
</tr>
<tr>
<td>$1$</td>
<td>A column vector with all the elements are 1, $(1 \in \mathbb{R}^N)$</td>
</tr>
<tr>
<td>$I$</td>
<td>Identity matrix</td>
</tr>
<tr>
<td>$Tr(\cdot)$</td>
<td>Trace operator</td>
</tr>
<tr>
<td>$t$</td>
<td>The iterative step in the DUGE algorithm</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>A regularization parameter of a penalty term</td>
</tr>
<tr>
<td>$f_i$</td>
<td>The row vector of i-th point in cluster indicator matrix</td>
</tr>
<tr>
<td>$F$</td>
<td>The cluster indicator matrix</td>
</tr>
<tr>
<td>$\Omega(F)$</td>
<td>The global discriminative information</td>
</tr>
<tr>
<td>$\lambda_{\infty}$</td>
<td>A large enough value for keeping the $c$ smallest eigenvalue of $L$ equal to zero</td>
</tr>
<tr>
<td>$\xi$</td>
<td>A regularization parameter for trade-off of global discriminative information</td>
</tr>
<tr>
<td>$\hat{X}_i$</td>
<td>The vector form of 2D image data</td>
</tr>
<tr>
<td>$\mu$</td>
<td>The mean of all data points in $\hat{X}$</td>
</tr>
</tbody>
</table>

Table 3. Notations
The Discriminative Unsupervised 2D Dimensionality Reduction with Graph Embedding (DUGE) algorithm \[1\] is proposed by performing dimensionality reduction, similarity matrix learning and spectral clustering simultaneously. To avoid overfitting, the DUGE also incorporates the global discriminative information for more robust results by minimizing the covariance of the within-cluster scatter matrix and maximizing the covariance of the between-cluster scatter matrix. Moreover, an optimization iteration algorithm is developed to obtain the desirable projection matrices $U$ and $V$.

Firstly, the problem is formulated as

$$\min_{P,U,V,F} \left( \sum_{i,j=1}^{N} \| U^T X_i V - U^T X_j V \|_F^2 + \gamma P_{ij}^2 + \lambda_\infty \| f_i - f_j \|_2^2 P_{ij} \right) + \xi \Omega(F)$$

s.t. $U^T U = I, V^T V = I, P1 = 1, 0 \leq P_{ij} \leq 1, 1 \leq i, j \leq N, F^T F = I,$

where $\Omega(F) = Tr[F^T H F - F^T X^T (X X^T + \mu I)^{-1} X^T F], \xi \geq 0$ is a regularization parameter.

In problem (1), \( \min_{P,U,V,F} \left( \sum_{i,j=1}^{N} \| U^T X_i V - U^T X_j V \|_F^2 + \gamma P_{ij}^2 + \lambda_\infty \| f_i - f_j \|_2^2 P_{ij} \right) \) is used to obtain the desirable similarity matrix by performing matrix multiplication in both the row-direction and the column-direction to preserve both the intrinsic neighborhood geometry of the data samples and the global geometry. In particular to $P_{ij}$, it is a probability of data points between $X_i$ and $X_j$, which is commonly used to preserve neighbor identities based on Stochastic Neighbor Embedding(SNE) algorithm \[10\], spectral clustering is adopted as the term $\sum_{i,j=1}^{N} \lambda_\infty \| f_i - f_j \|_2^2 P_{ij}$ in the objective function. Lastly, $\xi \Omega(F)$ represents the global discriminative information learning of data which can effectively avoid overfitting and clustering performance degradation lead by over-emphasis of local structures of data.

**Thesis 1.1:** I provided a proof for the convergence of the “Discriminative Unsupervised 2D Dimensionality Reduction with Graph Embedding”algorithm. \[13\]

**Theorem 1** The inequality $\sum_{i,j=1}^{N} \left( \| U^{t+1}^T X_i V^{t+1} - U^{t+1}^T X_j V^{t+1} \|_F^2 + \gamma P_{ij}^{t+1} + \lambda_\infty \| f_i^{t+1} - f_j^{t+1} \|_2^2 P_{ij}^{t+1} \right) + \xi \Omega(F^{t+1}) \leq \sum_{i,j=1}^{N} \left( \| U^{t}^T X_i V^{t} - U^{t}^T X_j V^{t} \|_F^2 P_{ij} + \gamma P_{ij}^{t} + \lambda_\infty \| f_i^{t} - f_j^{t} \|_2^2 P_{ij}^{t} \right) + \xi \Omega(F^{t})$ in the iterative process holds, where $t$ denotes the iteration step.

**Proof:** With the fixed $P,F$ and $U$, the objective function of (1) is transformed convex optimization problem \[3\] which can be solved by taking the derivative to 0 and the global solution of $V$ is obtained. In the same way, we fixed $P,F$ and $V$, the objective function of (1) is transformed to \[2\] which is obtained. Similarly, if $P,U$ and $V$ are fixed, (1) is transformed into the form of $\min_{F \in \mathbb{R}^{N \times c}, F^T F = I} Tr[F^T (2\lambda_\infty L + \xi R) F]$ and then $F$ is solved by setting the derivative with respect to $F$ to 0. Finally, with fixed $U,V,F$, $P$ is easily obtained by solving (4) according to the method proposed in \[17\].

After the $t$-th iteration, the updated $U,V,P$ and $F$ are denoted as $U^t$, $V^t$, $P^t$ and $F^t$ respectively. Similarly, they are denoted as $U = U^{t+1}$, $V = V^{t+1}$, $P = P^{t+1}$ and $F = F^{t+1}$ in the next iteration.

If we fixed $P^t, V^t$ and $F^t$, the following inequality is obtained:

$$\sum_{i,j=1}^{N} \left( \| U^{t+1}^T X_i V^t - U^{t+1}^T X_j V^t \|_F^2 P_{ij} + \gamma P_{ij}^{t+2} + \lambda_\infty \| f_i^{t} - f_j^{t} \|_2^2 P_{ij}^{t} \right) + \xi \Omega(F^{t})$$

$$\leq \sum_{i,j=1}^{N} \left( \| U^{t}^T X_i V^{t} - U^{t}^T X_j V^{t} \|_F^2 P_{ij} + \gamma P_{ij}^{t} + \lambda_\infty \| f_i^{t} - f_j^{t} \|_2^2 P_{ij}^{t} \right) + \xi \Omega(F^{t}). \quad (5)$$
Algorithm 1: The optimization algorithm of DUGE

**Data:** Data points $X_1, X_2, \ldots, X_N$, the parameters $k, c, u, v, \xi$ and $\lambda_\infty$.

**Result:** Projection matrices $U \in \mathbb{R}^{n \times u}$ and $V \in \mathbb{R}^{n \times v}$.

Initialize column $i$ of $P$, $i = 1, \ldots, N$, by solving the optimization problem

$$
\min_{P_{1=1,0 \leq P_{ij} \leq 1}} \sum_{j=1}^{N} \|X_i - X_j\|^2 F P_{ij} + \gamma P_{ij}^2.
$$

The initial matrices of $V$ and $U$ are set as an arbitrary column orthogonal matrix;

Set $t = 0$;

repeat

1. Update $L^t = D^t - \frac{P^t P^T + P^T P^t}{2}$, where $D^t \in \mathbb{R}^{N \times N}$ is a diagonal matrix with the $i$-th diagonal element as $\sum_j (P^t_{ij} + P^t_{ji})/2$;

2. Update $F^t$, whose columns are the $c$ eigenvectors of $(2\lambda_\infty L^t + \xi R)$ corresponding to its $c$ smallest eigenvalues, where $R = H - \tilde{X}^T \tilde{X}^T + \mu I$;

3. Update $U^t$, whose columns are the $u$ eigenvectors of $W^u_t$ corresponding to the $u$ smallest eigenvalues in equation

$$
\min_{U^T U = I} Tr(U^T W^v U).
$$

4. Update $V^t$, whose columns are the $v$ eigenvectors of $W^u_t$ corresponding to the $v$ smallest eigenvalues in equation

$$
\min_{V^T V = I} Tr(V^T W^v V).
$$

5. Update the $i$-th column of $P^t$, $i = 1, \ldots, N$, by solving

$$
\min_{P_{T1=1,0 \leq P_{ij} \leq 1,1 \leq j \leq N}} \|P_i + \frac{1}{2\gamma} d_i\|^2_2,
$$

where $d_i \in \mathbb{R}^{N \times 1}$ is a vector with the $j$-th element is $d_{ij} = d_{ij}^1 + d_{ij}^2$;

$t = t + 1$;

until Convergence;

Return the projection matrices $U$ and $V$. 

5
In the same way, if \( P^t, U^t \) and \( F^t \) are fixed, we have
\[
\sum_{i,j=1}^{N} \left( \| U^{tT} X_i V^{t+1} - U^{tT} X_j V^{t+1} \|^2_F P_{ij}^t + \gamma P_{ij}^{t+1} + \lambda_\infty \| f_i^t - f_j^t \|^2_F P_{ij}^t \right) + \xi \Omega(F^t)
\]
\[
\leq \sum_{i,j=1}^{N} \left( \| U^{tT} X_i V^t - U^{tT} X_j V^t \|^2_F P_{ij}^t + \gamma P_{ij}^t + \lambda_\infty \| f_i^t - f_j^t \|^2_F P_{ij}^t \right) + \xi \Omega(F^t). \tag{6}
\]
when \( U^t, V^t, F^t \) are fixed,
\[
\sum_{i,j=1}^{N} \left( \| U^{tT} X_i V^t - U^{tT} X_j V^t \|^2_F P_{ij}^t + \gamma P_{ij}^{t+1} + \lambda_\infty \| f_i^{t+1} - f_j^{t+1} \|^2_F P_{ij}^t \right) + \xi \Omega(F^t)
\]
\[
\leq \sum_{i,j=1}^{N} \left( \| U^{tT} X_i V^t - U^{tT} X_j V^t \|^2_F P_{ij}^t + \gamma P_{ij}^t + \lambda_\infty \| f_i^t - f_j^t \|^2_F P_{ij}^t \right) + \xi \Omega(F^t). \tag{7}
\]
With fixed \( P^t, V^t \) and \( U^t \), we obtain
\[
\sum_{i,j=1}^{N} \left( \| U^{tT} X_i V^t - U^{tT} X_j V^t \|^2_F P_{ij}^t + \gamma P_{ij}^t + \lambda_\infty \| f_i^t - f_j^t \|^2_F P_{ij}^t \right) + \xi \Omega(F^t)
\]
\[
\leq \sum_{i,j=1}^{N} \left( \| U^{tT} X_i V^t - U^{tT} X_j V^t \|^2_F P_{ij}^t + \gamma P_{ij}^t + \lambda_\infty \| f_i^t - f_j^t \|^2_F P_{ij}^t \right) + \xi \Omega(F^t). \tag{8}
\]
Additionally, consider the following two inequalities
\[
\sum_{i,j=1}^{N} \| U^{tT+1} X_i V^{t+1} - U^{t+1T} X_j V^{t+1} \|^2_F P_{ij}^{t+1}
\]
\[
\leq \sum_{i,j=1}^{N} \| U^{tT} X_i V^t - U^{tT} X_j V^t \|^2_F P_{ij}^t, \tag{9}
\]
and
\[
\sum_{i,j=1}^{N} \left( \| f_i^{t+1} - f_j^{t+1} \|^2_F P_{ij}^{t+1} \right) + \xi \Omega(F^{t+1}) \leq \sum_{i,j=1}^{N} \left( \| f_i^t - f_j^t \|^2_F P_{ij}^t \right) + \xi \Omega(F^t) \tag{10}
\]
are satisfied. Sum over the formulas from (5) to (10), we can get
\[
\sum_{i,j=1}^{N} \left( \| U^{tT+1} X_i V^{t+1} - U^{t+1T} X_j V^{t+1} \|^2_F P_{ij}^{t+1} + \gamma P_{ij}^{t+1} \right) + \lambda_\infty \| f_i^{t+1} - f_j^{t+1} \|^2_F P_{ij}^{t+1} \right)
\]
\[
\leq \sum_{i,j=1}^{N} \left( \| U^{tT} X_i V^t - U^{tT} X_j V^t \|^2_F P_{ij}^t + \gamma P_{ij}^t + \lambda_\infty \| f_i^t - f_j^t \|^2_F P_{ij}^t \right) + \xi \Omega(F^t). \tag{11}
\]
That is, Theorem 1 is proved. As a result, the objective function of (11) in the iteration process converges to the minimum value and the desirable matrices \( U \) and \( V \) are obtained when Algorithm 1 is executed.
4.1.2 Numerical Evaluation

**Thesis 1.2:** I showed that DUGE algorithm has better performance than others in various scenarios. [11]

I evaluated and compared the performance of DUGE algorithm with several 2-dimensional unsupervised dimensionality reduction algorithms by the extensive experiments on four benchmark data sets. The clustering performance is evaluated by K-means in terms of Accuracy (ACC) and Normalized Mutual Information (NMI) [2]. Table 4 shows the Accuracy (ACC) measurements of diverse unsupervised dimensionality reduction methods.

<table>
<thead>
<tr>
<th>Data set</th>
<th>JAFFE</th>
<th>FERET</th>
<th>AT&amp;T</th>
<th>Coil20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>53.90</td>
<td>33.10</td>
<td>58.21</td>
<td>59.07</td>
</tr>
<tr>
<td>RPCAOM</td>
<td>57.00</td>
<td>33.83</td>
<td>61.26</td>
<td>60.32</td>
</tr>
<tr>
<td>(2D)PCA</td>
<td>53.40</td>
<td>33.58</td>
<td>59.35</td>
<td>59.65</td>
</tr>
<tr>
<td>(2D)2PCA</td>
<td>54.97</td>
<td>33.63</td>
<td>59.67</td>
<td>60.38</td>
</tr>
<tr>
<td>I(2D)2PCA</td>
<td>53.89</td>
<td>33.30</td>
<td>59.04</td>
<td>59.81</td>
</tr>
<tr>
<td>(2D)LPP</td>
<td>55.30</td>
<td>33.56</td>
<td>61.92</td>
<td>60.85</td>
</tr>
<tr>
<td>DRASL</td>
<td>60.00</td>
<td>35.9</td>
<td>63.50</td>
<td>62.39</td>
</tr>
<tr>
<td>DUGE</td>
<td>59.50</td>
<td><strong>38.57</strong></td>
<td><strong>65.12</strong></td>
<td><strong>66.51</strong></td>
</tr>
</tbody>
</table>

Table 4. Clustering result in terms of accuracy(%)  

Table 5 shows the Normalized Mutual Information (NMI) measurements of several unsupervised dimensionality reduction methods.

<table>
<thead>
<tr>
<th>Data set</th>
<th>JAFFE</th>
<th>FERET</th>
<th>AT&amp;T</th>
<th>Coil20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>55.44</td>
<td>63.08</td>
<td>79.01</td>
<td>73.04</td>
</tr>
<tr>
<td>RPCAOM</td>
<td>58.59</td>
<td>64.52</td>
<td>80.93</td>
<td>74.73</td>
</tr>
<tr>
<td>(2D)PCA</td>
<td>54.54</td>
<td>63.17</td>
<td>79.43</td>
<td>74.09</td>
</tr>
<tr>
<td>(2D)2PCA</td>
<td>54.93</td>
<td>63.46</td>
<td>78.87</td>
<td>72.78</td>
</tr>
<tr>
<td>I(2D)2PCA</td>
<td>54.91</td>
<td>63.34</td>
<td>79.16</td>
<td>74.11</td>
</tr>
<tr>
<td>(2D)LPP</td>
<td>56.23</td>
<td>63.60</td>
<td>81.05</td>
<td>74.10</td>
</tr>
<tr>
<td>DRASL</td>
<td>62.66</td>
<td>65.47</td>
<td>81.98</td>
<td>76.02</td>
</tr>
<tr>
<td>DUGE</td>
<td>59.73</td>
<td><strong>66.70</strong></td>
<td><strong>83.87</strong></td>
<td><strong>76.35</strong></td>
</tr>
</tbody>
</table>

Table 5. Clustering result in terms of normalized mutual information(%)  

The results show that DUGE algorithm achieved remarkable clustering performance than others except for DRASL on JAFFE data set.
4.2 Characterization and Prediction models for MapReduce applications

Thesis Group 2 Characterization of MapReduce applications and forecasting models for the resource usage of MapReduced applications [J4, J2, J3, C1, C2]

4.2.1 Characterization of MapReduce applications

Thesis 2.1 [J4]: I showed that

- the autocorrelation coefficient of CPU usage of CPU-intensive MapReduce application is positive high, the correlation coefficient between CPU usage and memory usage as well;

- the autocorrelation coefficients of read rate of read-intensive MapReduce applications are positive high, whereas the autocorrelation coefficients of write rate are very low (In other words, we can say there is randomness in the values of write rate). Additionally, the correlation coefficient of read rate and memory usage is positive and at least moderate and the correlation coefficient of write rate and memory usage is very low;

- the autocorrelation coefficient of write rate of write-intensive MapReduce application is positive high, the correlation coefficient of write rate and memory usage is positive and at least low, the correlation coefficient of read rate and memory usage is negative and at least low;

- the autocorrelation coefficient of write rate of read/write-intensive MapReduce application is positive high, the correlation coefficient of read rate and either CPU usage or memory usage is positive and at least low, the correlation coefficient of CPU usage and memory usage is at most low.

To characterize different types of MapReduce applications, I have performed the analysis of MapReduce applications. Particularly, the subjects of characterization are resource usage parameters, such as CPU usage, memory usage, read rate and write rate.

The non-randomness and identified the lag number for each usage parameter by observing the autocorrelation function plot (ACF) and autocovariance function plot was investigated [8]. The ACF plot and autocovariance plot of Pi application (CPU-intensive) and Terasort application (read/write-intensive) are plotted in Figure 1 and 2, respectively.

The top half panel of Figure 1 shows that both the CPU usage and the memory usage reveal non-randomness because of autocorrelations, denoted by the circle, violating the dashed lines and being statistically significant for lags up to 100. The filled triangle point-up marks the autocorrelation of the CPU usage and its lag 1 values at 0.983. The autocorrelation for the memory usage is 0.932. These values show highly positive relevance between the CPU usage and its lag 1 values as well as between the memory usage and its lag 1 values. Meanwhile, the autocovariance of CPU usage and that of the memory usage with their lag 1 values are 1516.341 and 16.877, respectively, which shows the significant non-zero quantity.

On the bottom left panel of Figure 1, although there is high autocorrelation (0.732) between read rate and its lag 1 values, the corresponding autocovariance is 0.029. This result indicates that the read rate is non-random. The same can be observed with the write rate.

Turning to Tersort application, the significant non-randomness of the CPU usage, the memory usage, the read rate and the write rate are exposed in Figure 2. On ACF plot of each resource usage parameter, most of the circles which correspond to each lag, are far from the dashed lines and statistically significant for lags up to 100. The corresponding autocovariance
Figure 1. Autocorrelation plot and autocovariance plot for Pi application

plot shows the unstandardized variation between each lag with usage parameter itself. The filled triangle point up denotes autocorrelation or autocovariance between time series data and its lag 1 values. When compared with Figure 1, the autocovariance plots of the read rate and the write rate revealed the salient difference between Terasort application and Pi application. The autocovariance of Terasort, denoted by the filled triangle point up, between read rate variable and its lag 1 values, shows the strong dependency while that is not so in the case of Pi. The write rate has the similar characteristics with that of the read rate.
The correlation matrices for Pi application (CPU-intensive), Wordcount (read-intensive), Teragen (write-intensive) and Terasort (read/write-intensive) are showed in Table 6 to Table 9.

From the correlation matrix (Tables 6, 7, 8, and 9), it can be observed that (1) MapReduce applications with different resource intensive characteristics do not exhibit the similar characteristics on correlation and autocorrelation, and (2) MapReduce applications with the same resource-intensive type showed extremely similar signatures on correlation and autocorrelation.
MapReduce applications with different resource intensive characteristics show that:

- the exception is a high value associated to the autocorrelation coefficient of the memory usage because the investigated MapReduce applications reserve a memory to store and process data;
- there is no obvious relationship between the write rate and others resource usage parameters in the CPU-intensive applications and the read-intensive applications, while other two classes of applications present strong correlation. One possible explanation is that the write rate plays an insignificant role in these applications, e.g. Pi (CPU-intensive) and Wordcount (read-intensive);
- the correlation coefficient between CPU usage and memory usage of CPU-intensive applications is very high while such metric in read/write-intensive ones is very low (almost insignificant).

MapReduce applications with the same resource-intensive type showed the extremely similar signatures on correlation and autocorrelation. Based on some common signatures about correlation coefficient and autocorrelation of resource usage parameters, we can identify the resource-intensive categorization to which the MapReduce application belongs. This is the highlight of this work. Some common signatures could be summarized to identify the categorization of MapReduce applications. For convention, the categorized thresholds of the correlation coefficient are defined in Table 10.

According to the different resource-intensive types, the distributions of these correlation coefficients and autocorrelation are exhibited in Figure 3 to Figure 6. In these figures, all tested MapReduce applications perform the same characteristics on the perfect autocorrelation of memory usage and strong positive autocorrelation of read rate. Note that the different types of dashed lines present the threshold level in Table 10.

In Figure 3 the correlation coefficient between CPU usage and memory usage as well as autocorrelation of CPU usage shows an extremely high value which is larger than 0.9. Meanwhile, read rate and write rate show almost no relationship between them because the absolute value of correlation coefficient between them is less than 0.1. Some signatures for CPU-intensive MapReduce applications are summarized as:

<table>
<thead>
<tr>
<th>Pi Application</th>
<th>CPU usage</th>
<th>memory usage</th>
<th>read rate</th>
<th>write rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory usage</td>
<td>0.957</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read rate</td>
<td>-0.122</td>
<td>-0.156</td>
<td></td>
<td></td>
</tr>
<tr>
<td>write rate</td>
<td>0.079</td>
<td>0.067</td>
<td>-0.02</td>
<td></td>
</tr>
<tr>
<td>lag1 autocorrelation</td>
<td>0.984</td>
<td>0.933</td>
<td>0.732</td>
<td>-0.025</td>
</tr>
</tbody>
</table>

Table 6. Correlation matrix of Pi application

<table>
<thead>
<tr>
<th>Wordcount Application</th>
<th>CPU usage</th>
<th>memory usage</th>
<th>read rate</th>
<th>write rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory usage</td>
<td>0.142</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read rate</td>
<td>0.228</td>
<td>0.279</td>
<td></td>
<td></td>
</tr>
<tr>
<td>write rate</td>
<td>-0.002</td>
<td>0.005</td>
<td>-0.169</td>
<td></td>
</tr>
<tr>
<td>lag1 autocorrelation</td>
<td>0.627</td>
<td>1</td>
<td>0.61</td>
<td>-0.018</td>
</tr>
</tbody>
</table>

Table 7. Correlation matrix of Wordcount application
<table>
<thead>
<tr>
<th>Teragen Application</th>
<th>CPU usage</th>
<th>memory usage</th>
<th>read rate</th>
<th>write rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory usage</td>
<td>-0.396</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read rate</td>
<td>0.077</td>
<td>-0.436</td>
<td></td>
<td></td>
</tr>
<tr>
<td>write rate</td>
<td>0.007</td>
<td>0.238</td>
<td>-0.415</td>
<td></td>
</tr>
<tr>
<td>lag1 autocorrelation</td>
<td>0.707</td>
<td>1</td>
<td>0.822</td>
<td>0.781</td>
</tr>
</tbody>
</table>

Table 8. Correlation matrix of Teragen application

<table>
<thead>
<tr>
<th>Terasort Application</th>
<th>CPU usage</th>
<th>memory usage</th>
<th>read rate</th>
<th>write rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory usage</td>
<td>-0.089</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>read rate</td>
<td>0.046</td>
<td>-0.269</td>
<td></td>
<td></td>
</tr>
<tr>
<td>write rate</td>
<td>0.048</td>
<td>-0.073</td>
<td>-0.495</td>
<td></td>
</tr>
<tr>
<td>lag1 autocorrelation</td>
<td>0.692</td>
<td>0.995</td>
<td>0.893</td>
<td>0.834</td>
</tr>
</tbody>
</table>

Table 9. Correlation matrix of Terasort application

Figure 3. Correlation coefficient of CPU-intensive application

- the autocorrelation coefficient of CPU usage is positive high,
- the correlation coefficient of CPU usage and memory usage is positive high.

In Figure 4, read-intensive applications perform similar correlation characteristics on three pairs of variables: (memory usage, read rate), (read rate, write rate), (memory usage, write rate) and autocorrelation of write rate. Therefore, read-intensive MapReduce application showed the signatures as:

- the autocorrelation coefficient of read rate is positive high,
- the autocorrelation coefficient of write rate is very low (In other words, we can say there is randomness in the values of write rate),
- the correlation coefficient of read rate and memory usage is positive and at least moderate,
<table>
<thead>
<tr>
<th>Threshold</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 0.1</td>
<td>Very low value presents no relationship</td>
</tr>
<tr>
<td>[0.1, 0.3)</td>
<td>Low value presents weak relationship</td>
</tr>
<tr>
<td>[0.3, 0.5)</td>
<td>Moderate value presents moderate relationship</td>
</tr>
<tr>
<td>[0.5, 1]</td>
<td>High value presents strong relationship</td>
</tr>
</tbody>
</table>

Table 10. Categorized threshold of correlation coefficient

Figure 4. Correlation coefficient of read-intensive application

- the correlation coefficient of write rate and memory usage is very low.

Figure 5. Correlation coefficient of write-intensive application
In Figure 5, the autocorrelations of write rate and read rate show the strong positive relevance. The correlation between read rate and write rate performs significantly negative relevance. Thus, some signatures of write-intensive MapReduce application are summarized as:

- the autocorrelation coefficient of write rate is positive high,
- the correlation coefficient of write rate and memory usage is positive and at least low,
- the correlation coefficient of read rate and memory usage is negative and at least low.

Read/write-intensive application (Terasort) shows its correlation coefficients and autocorrelation in Figure 6. Their values in Figure 6 are similar to those in Figure 5. However, correlation direction of them between memory usage and write rate is rather opposite. Thus, the signatures for read/write-intensive MapReduce application are:

- the autocorrelation coefficient of write rate is positive high,
- the correlation coefficient of read rate and either CPU usage or memory usage is positive and at least low,
- the correlation coefficient of CPU usage and memory usage is at most low.

4.2.2 Linear Regression Models

**Thesis 2.2:** I have established multiple linear regression models for seven benchmark MapReduce applications and revealed the minimal number of samples for stable modeling resource dependency pattern as follows [J2]:

- the samples collected at least in (2500, 3800, 2200, 2750, 2450, 3000, 12400) seconds are needed to establish the CPU usage model of applications (Wordcount, Wordmean, Wordmedian, Grep, Pi, Teragen, Terasort), respectively;
• the samples collected at least in (900, 900, 1500, 2600, 3100, 8050) seconds are needed to establish the memory usage model of applications (Wordcount, Wordmean, Wordmedian, Grep, Pi, Teragen, Terasort), respectively;

• the samples collected at least in (3500, 3250, 2900, 3800, 2500, 3200, 13300) seconds are needed to establish the read rate model of applications (Wordcount, Wordmean, Wordmedian, Grep, Pi, Teragen, Terasort), respectively;

• the samples collected at least in (3050, 2150, 3400, 3700, 2750, 2750, 13300) seconds are needed to establish the write rate model of applications (Wordcount, Wordmean, Wordmedian, Grep, Pi, Teragen, Terasort), respectively.

To establish multiple linear regression models, I proposed the following steps:

• step 1: to reconstruct performance dataset by incorporating the most significant lagged variable of each usage parameter into the original one in accordance with the largest partial autocorrelation coefficient.

• step 2: to select the necessary feature collection by removing collinearity variable, choosing best-subset, and adding useful interaction term.

• step 3: to establish resource dependency prediction models using multiple linear regression methods for each usage parameter of seven benchmark MapReduce applications.

• step 4: to collect the statistical metrics (residual standard error ($RSE$) and $R^2$) of each model for evaluating model fitting quality.

To illustrate the fit quality of prediction models, the statistical metrics of intensive-resource usage models are presented in Table 11.

<table>
<thead>
<tr>
<th>Application</th>
<th>Response Variable</th>
<th>RSE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordCount</td>
<td>CPU usage</td>
<td>13.41</td>
<td>39.5%</td>
</tr>
<tr>
<td></td>
<td>Read rate</td>
<td>3.70</td>
<td>38.9%</td>
</tr>
<tr>
<td>WordMean</td>
<td>CPU usage</td>
<td>16.25</td>
<td>36.1%</td>
</tr>
<tr>
<td></td>
<td>Read rate</td>
<td>2.73</td>
<td>56.7%</td>
</tr>
<tr>
<td>WordMedian</td>
<td>CPU usage</td>
<td>21.37</td>
<td>25.6%</td>
</tr>
<tr>
<td></td>
<td>Read rate</td>
<td>2.59</td>
<td>57.8%</td>
</tr>
<tr>
<td>Grep</td>
<td>CPU usage</td>
<td>11.86</td>
<td>55%</td>
</tr>
<tr>
<td></td>
<td>Read rate</td>
<td>2.64</td>
<td>64.4%</td>
</tr>
<tr>
<td>Terasort</td>
<td>CPU usage</td>
<td>9.41</td>
<td>47.8%</td>
</tr>
<tr>
<td></td>
<td>Read rate</td>
<td>3.19</td>
<td>90.4%</td>
</tr>
<tr>
<td></td>
<td>Write rate</td>
<td>6.39</td>
<td>83.7%</td>
</tr>
<tr>
<td>Teragen</td>
<td>Write rate</td>
<td>6.56</td>
<td>53.5%</td>
</tr>
<tr>
<td>Pi</td>
<td>CPU usage</td>
<td>7</td>
<td>96.8%</td>
</tr>
</tbody>
</table>

Table 11. Statistical measurements of intensive-resource usage model

From Table 11, the prediction model of CPU-intensive application (Pi) achieves the highest percentage of explained variance (96.8%), whereas read rate and write rate prediction models of read/write intensive application (Terasort) acquired the best prediction quality (90.4%...
and 83.7%). Read rate models of read-intensive applications (Grep, WordMedian, WordMean, WordCount) achieved the similar higher goodness of fit (64.4%, 57.8%, 56.7%, 38.9%). Write rate models of write-intensive applications (Teragen) got 53.5% \( R^2 \) value which represents a higher goodness of fit. Except CPU-intensive application, others showed at least moderate fit quality on CPU usage models. These metrics indicates that intensive-resource usage models of MapReduce application could achieve better goodness of fit than other non-intensive resource ones.

A common characteristic for all regression models is the positive dependency between resource usage parameters and its lagged variable. However, except for these common dependencies, there still have some particular dependent characteristics for each types of application, which are summarized as below:

- CPU-intensive application: the best fit quality on CPU usage model, the strongest positive dependency between current CPU usage and its lagged variable;
- read-intensive application: the better fit quality on read rate, the moderate positive dependency between read rate and their lagged variable;
- read/write intensive application: the best fit quality on both read and write rate models, the strong positive dependency of read rate and write rate between them and their lagged variables; and the strong negative dependency between read and write rate;
- write-intensive application: the best goodness of fit on write rate model, the high positive dependency between write rate and its lagged variable.

To evaluate the convergence of estimated coefficient and statistical metrics as sampling time constantly increasing, I formulated the changing rate for estimated coefficient and statistical metrics in equation (12, 14 and 13). The changing rate of the estimated coefficient is denoted by .

\[
\epsilon_{\beta_j} = \left| \frac{(\hat{\beta}_j - \beta_j)}{|\beta_j|} \right| \tag{12}
\]

where the molecular is the absolute difference between estimated coefficients \( \hat{\beta}_j \) and approximately true coefficients \( \beta_j \) (the estimated coefficients of regression models trained on the whole dataset), the denominator is the absolute value of approximately true coefficients.

Similarly, the changing rate of \( \text{RSE} \) and \( \text{R}^2 \) are denoted by

\[
\epsilon_{\text{RSE}} = \left| \frac{(RSE_{real} - RSE_{fitted\_model})}{|RSE_{real}|} \right| \tag{13}
\]

where \( RSE_{fitted\_model} \) indicates residual standard error of fitted models and \( RSE_{real} \) represents RSE of regression models trained on the whole dataset, and

\[
\epsilon_{\text{R}^2} = \left| \frac{(R^2_{real} - R^2_{fitted\_model})}{|R^2_{real}|} \right| , \tag{14}
\]

where \( R^2_{fitted\_model} \) indicates \( R^2 \) of fitted models and \( R^2_{real} \) represents \( R^2 \) of regression models trained on the whole dataset using K-fold cross validation approach. The threshold 0.1 means that the corresponding measurement reaches stability as the changing rate is less than 10%.

Based on the changing rate equations (12, 14 and 13), I conducted the stepwise experiment to obtain the minimal sampling time by comparing the changing rates of each metric with the
threshold (0.1). The minimal number of samples for stable modeling are summarized in Figure 7 (estimated coefficients 7a and statistic metrics 7b) accordingly.

Figure 7a reveals that the regression models of Terasort needs the largest number of samples to reach stability while Pi is opposite. The rest applications require similar minimal sampling time. Comparing to others, memory usage models needs the shortest sampling time to reach stability. The experimental results show that stable modeling for various resource-intensive types of applications presents distinct requirements for sampling time.

![Minimum Sampling time of Estimated Coefficients](image1)

![Minimum Sampling time of Statistical Metrics](image2)

(a) Estimated coefficients  
(b) Statistical metrics

Figure 7. The minimal sampling time for stable modeling

From Figure 7b, the minimal numbers of samples for statistic metrics showed the smaller values than the estimated coefficients. Read/write intensive applications (Terasort) need the longest sampling time while Pi application shows the smallest sampling time for stability.

### 4.2.3 LSTM models to predict the resource usage of MapReduce applications

**Thesis 2.3:** I have established forecasting models based on long short-term recurrent neural networks (LSTM) for the resource usage of four representative MapReduce applications. I have showed that LSTM models of intensive-resource usage achieved better prediction accuracy than multiple linear regression models as the training data is sufficient. Also I revealed that LSTM models are more sensitive to sample size than multiple linear regression models.

Long Short Term Memory networks (LSTM) is an extension version of recurrent neuron network (RNN) being able to learn long-term dependencies. Prior to establish LSTM model, the selection of Hyperparameters is an essential step. In this study, six Hyperparameters (epoch size, batch size, neuron number, time steps, dropout rate, hidden layer number) are tuned in order. I have proposed an approach to find Hyperparameters.

To compare prediction performance of distinct methods, I compared NRMSE value of each model for four MapReduce applications in Figure 8. Note that the performance baseline NRMSE is calculated to be the threshold of forecasting effectiveness.

Notably, the prediction accuracy of LSTM model was significantly higher than the corresponding regression model. For memory usage prediction, Figure 8a shows that the forecasting models of either multivariate LSTM or multiple linear regression is meaningless for data-intensive
application yet is only sufficient to CPU-intensive-only applications. Figure 8c shows that read-intensive application presents significant improvement in prediction accuracy than either the performance baseline or regression model. Two-hidden-layer LSTM model cannot significantly increase prediction accuracy. Figure 8d presents that read/write intensive and write-intensive applications can effectively predict write rate for relevant applications. The common characteristic of Teragen and Terasort is write-intensive property.

Figure 9. Sensitivity comparison of CPU usage forecasting models
Figure 9 shows that the sample size has less impact on the prediction accuracy of CPU usage regression model. However, LSTM models of CPU usage are very sensitive to the small sample size. The experimental results show that LSTM models are sensitive to a sample size and need longer training time to reach a stable state [J3], while multiple linear regression models are insensitive to sample size for most of applications. It is observed that LSTM forecasting models of intensive resource usage achieve more accurate prediction when the sample size is sufficient.

**Thesis 2.4:** I have proposed a two-phase modeling method that is effectively to cope with the underfitting problem for the resource usage prediction of the specific application (Terasort). [J3]

Although LSTMs methods have achieved remarkable prediction accuracy on forecasting usage parameters, the overfitting [9] and underfitting problems are often seen in machine learning process. Thus, I have examined these two problems for each LSTM model and found LSTM models of Terasort application have seriously underfitting. For Terasort, the consumed time is 3728 seconds for map phase and 8800 seconds for pure reduce phase. It can be assumed that distinct computing characteristics were existing in two separated phases. For this reason, I proposed a two-phase modeling approach to alleviate underfitting problem for Terasort application, the corresponding evaluation as well.

I modeled usage parameters of Terasort for two different phases: map phase and pure reduce phase, respectively. I divided dataset into map and reduce phase sample. Then, I applied the Hyperparameters learning and prediction algorithms again. The comparison of train and test NRMSE for forecasting models are depicted in Figure 10.

---

**Figure 10.** Overall modeling vs separated phase modelling
Figure 10 consists of three sub-figures: (a) prediction accuracy on overall sample dataset, (b) prediction accuracy on map phase (including pure map and mixed phase) sample dataset, (c) prediction accuracy on pure reduce phase sample dataset. Sub-figure (a) showed the heavy underfitting on all usage parameters prediction. In map phase models (sub-figure (b)), NRMSE of all usage parameters predictions show the removing of underfitting which represents the forecasting models are valuable. Particularly, models for read and write rate predictions achieved good fit performance and models for CPU usage had slightly overfitting. Moving to sub-figure (c), the test and train error of models for CPU usage prediction presents incredible similarity which represents good accuracy of the LSTM model.

As a consequence, the two-phase modeling approach for specific application can correctly reveal usage parameters pattern than overall modeling, such as Terasort with a long reduce phase.

**Thesis 2.5:** I have proposed a scheme to predict usage parameters of write-intensive applications based on the previously trained LSTM model and have showed that the proposed approach is accurate to predict the resource usage of write-intensive applications. [J3]

The main idea of this extensive study is to answer a question whether LSTM models trained on performance data from machine A can be applied to predict the usage parameters on machine B. The predicted values on machine B can be achieved by multiplying the prediction one of machine A by the difference ratio between two machines. The prediction accuracy of CPU usage and write rate is depicted in Figure 11. The result shows that the internal structure of the LSTM model is preserved [J3].

Figures 11a and 11b depict the prediction and the real value of the CPU usage and write rate of Teragen application. In Figure 11a, the extensive CPU usage prediction (blue dashed line) shows the similar values with the real value (solid red line). Figure 11b presents the comparisons between write rate prediction and real value for Teragen application. As can be seen, the write rate prediction by extensive model (blue dashed line) was close to real value (solid red line).
Therefore, it can be concluded that the extensive prediction of LSTM model is apparently useful for CPU usage and write rate prediction of write-intensive application.

5 Applications of results

The DUGE optimization algorithm proposed in the first thesis can be directly applied to extract salient and informative features from 2D unlabeled image data.

In thesis 2 group, the models quantify the resource usage of applications. The obtained models might be applied to allocate resource in Hadoop computing clusters.
Own Journal Publication


Own Conference Publication


References


