Multiple Learning for Regression in Big Data

Xiang Liu¹, Ziyang Tang¹, Huyunting Huang¹, Tonglin Zhang² and Baijian Yang¹*

¹Department of Computer and Information Technology, Purdue University
²Department of Statistics, Purdue University

{xiang35, tang385, huan1182, tlzhang, byang}@purdue.edu

Abstract

Regression problems that have closed-form solutions are well understood and can be easily implemented when the dataset is small enough to be all loaded into the RAM. Challenges arise when data is too big to be stored in RAM to compute the closed form solutions. Many techniques were proposed to overcome or alleviate the memory barrier problem but the solutions are often local optimal. In addition, most approaches require accessing the raw data again when updating the models. Parallel computing clusters are also expected if multiple models need to be computed simultaneously. We propose multiple learning approaches that utilize an array of sufficient statistics (SS) to address this big data challenge. This memory oblivious approach breaks the memory barrier when computing regressions with closed-form solutions, including but not limited to linear regression, weighted linear regression, linear regression with Box-Cox transformation (Box-Cox regression) and ridge regression models. The computation and update of the SS array can be handled at per row level or per mini-batch level. And updating a model is as easy as matrix addition and subtraction. Furthermore, multiple SS arrays for different models can be easily computed simultaneously to obtain multiple models at one pass through the dataset. We implemented our approaches on Spark and evaluated over the simulated datasets. Results showed our approaches can achieve closed-form solutions of multiple models at the cost of half training time of the traditional methods for a single model.

1 Introduction

Linear regression, weighted linear regression, linear regression with Box-Cox transformation (Box-Cox regression) and ridge regression have powered the society in many respects by modeling the relationship between a scalar response variable and explanatory variable(s). From housing price prediction to stock price prediction, and from face recognition to marketing analysis, the related applications span a wide spectrum [Naseem et al., 2010; Altay and Satman, 2005; Nghiep and Al, 2001; DeSarbo and Cron, 1988]. After entering the big data era, these regression models are still prevalent in academia and industry. Even more advanced models like XGBoost and deep learning have appeared, the regression models continue their impact in many fields [Chen and Guestrin, 2016; He et al., 2016]. With massive data, they can produce better performance. However, it is not easy to accomplish such task because of the memory barrier issue of the closed-form solutions and the accuracy and time efficiency issue of the iterative methods. Hence, it is of high value to propose new approaches that could preserve both pros of the methods above for linear regression, weighted linear regression, Box-Cox regression and ridge regression.

For linear regression, academia and industry resort to mainly two techniques, ordinary least squares (OLS) and the iterative methods. The OLS method is designed to calculate the closed-form solution [Kenney and Keeping, 1962]. By solving the normal equation, OLS could immediately derive the solution from the data. As normal equation consists of $(X^\top X)^{-1}$, if $X^\top X$ is singular, the normal equation will become unsolvable. One solution is to use generalized inverse [Dresden, 1920; Barata and Hussein, 2012; Ben-Israel and Greville, 2003]. Although OLS is time-wise efficient in deriving the closed-form solution, it also introduces the memory barrier issue that RAM-accommodable datasets are required to solve the equation. To overcome the issue of memory barrier, the distributed matrix could be applied to perform the calculation as a remedy [Moler, 1986]. But the time cost makes this algorithm infeasible nevertheless. Due to this reason, the applications of this technique is limited. And another technique, the iterative methods which include gradient descent, Newton’s method and Quasi-Newton’s method are commonly used. [Kiewel, 2001; Wedderburn, 1974].

Gradient descent, also known as steepest descent, targets to find the minimum of a function. It approaches the minimum by taking steps along the negative gradient of the function with a learning rate proportional to the gradient. It is more universal than OLS as the variations, such as mini-batch gradient descent and stochastic gradient descent, overcome the memory barrier issue by performing a calculation
in small batches instead of feeding all data into memory at once [Ruder, 2016]. But, gradient descent oscillates around the minimum region when the algorithm gets close to the minimum. And its asymptotic rate of convergence is inferior to many other iterative methods. If an easier approach to the minimum or higher asymptotic rate of convergence is demanded, Newton’s method is an alternative.

Newton’s method is a root-finding algorithm, utilizing the Taylor series. To find a minimum/maximum, it needs the knowledge of the second derivative. Unlike gradient descent, this strategy enables Newton’s method to approach the extrema/optimum more easily rather than oscillations. Besides, it has been proven that Newton’s method has the quadratic asymptotic rate of convergence, whereas, this algorithm is only faster than gradient descent when the Hessian matrix is known or easy to compute [Wedderburn, 1974]. Unfortunately, the expression for the second derivative is often complicated and intractable.

Quasi-newton methods, for instance, DFP, BFGS and L-BFGS, were proposed as alternatives to Newton’s method when the Hessian matrix is unavailable or too expensive to calculate [Davidon, 1978; Avriel, 2003; Malouf, 2002]. Instead of inverting the Hessian matrix in Newton’s method, quasi-newton methods build up an approximation for the inverse matrix to reduce the computational load. With this mechanism, quasi-newton methods are usually faster than Newton’s method in practice for large datasets. In linear regression, L-BFGS, a variation of BFGS, is one of the most widely used quasi-newton method [Zaharia et al., 2010]. Generally, L-BFGS outperforms gradient descent in linear regression.

Weighted linear regression is a more generalized version of linear regression by quantifying the importance of different observations [Myers and Myers, 1990]. A weighted version of OLS is designed to obtain the corresponding close-form solution. The iterative methods with slight modification are also applicable to weighted linear regression [Holland and Welsch, 1977].

For Box-Cox regression, it is linear regression with the response variable changed by Box-Cox transformation [Box and Cox, 1964]. The design philosophy of Box-Cox regression is to handle non-linearity between the response variable and explanatory variables by casting power transformation on the response variable. Naturally, approaches for linear regression are applicable to Box-Cox regression.

As linear regression is deficient in handling highly-correlated data, ridge regression is then proposed [Hoerl and Kennard, 1970]. The basic idea of ridge regression is to add a \( \ell_2 \) penalty term to the error sum of squares (SSE) cost function of linear regression [Hoerl and Kennard, 1970; Marquardt, 1970]. A constrained version of OLS can solve this problem, producing similar closed-form solution. The only difference is that the \((X^\top X)^{-1}\) component from OLS is substituted by \((X^\top X + \lambda I)^{-1}\), where \( \lambda \) is the coefficient of \( \ell_2 \) penalty, and \( I \) is the identity matrix. By means of \( I \), constrained OLS no longer has to deal with the singularity issue but the memory barrier issue from OLS remains. Gradient descent, Newton’s method and quasi-newton methods as well can be applied [Kiwiol, 2001; Wedderburn, 1974; Dennis and Moré, 1977].

Given the pros and cons of these regression models discussed above, some common drawbacks could be concluded: (i) OLS and its extended versions are difficult in handling the memory barrier issue. (ii) The iterative methods are time inefficient and require many iterations to well-train regression models. In addition to that, parameter tuning is inevitable under most conditions. It may probably take several days/weeks for large scale projects to accomplish the desired performance goals of models. For Box-Cox regression or ridge regression, the situation gets worse as multiple power or ridge parameters are usually applied to pick the best one, which, of course, also multiply the time cost [Pedregosa et al., 2011].

In order to preserve the pros of OLS based approaches that produce closed-form solutions and the iterative methods that overcome the memory barrier issues, we propose multiple learning approaches that utilize sufficient statistics (SS). The main contributions of our algorithms are summarized as below:

- We introduced a SS array which can be computed at per low or per mini-batch level for calculating closed-form solutions.
- Once the closed-form solutions are obtained, the optimums are found, i.e., the prediction performance is at least as good as OLS.
- With SS, the datasets are only scanned once, the time performance is efficiently improved in contrast to the iterative methods.
- As multiple SS arrays for different models can be computed simultaneously, multiple models are obtainable at one pass through the dataset.

The remainder of the paper is structured as follows. In Section 2, a closely relevant background is introduced; In Section 3, the multiple learning approaches for different regression models are presented; In Section 4, experiments and analysis are described and discussed. In Section 5, the work of this paper is concluded.

## 2 Background

For regression analysis, not only the estimators of the regression model coefficients \( \beta \) are calculated, but also the estimators of variance \( \sigma^2 \) and the variance-covariance matrices \( V(\beta) \), are computed for significant test. Thus, for ease of presentation, necessary notions and notations closely relevant to linear regression, weighted linear regression, Box-Cox regression and ridge regression are explained.

### 2.1 Linear Regression

Assume the dataset contains \( n \) observations each of which has \( p - 1 \) features. Consider a linear regression model

\[
y = X\beta + \varepsilon
\]

(1)

where \( y = (y_1, y_2, \ldots, y_n)^\top \) is a \( n \times 1 \) vector of the response variables, \( X = (x_{11}, x_{21}, \ldots, x_{n1})^\top \) is a \( n \times p \) matrix of explanatory variables, \( \beta = (\beta_0, \beta_1, \ldots, \beta_{p-1})^\top \)
is a $p \times 1$ vector of regression coefficient parameters, and $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_n)^T$ is the error term which is a $n \times 1$ vector following the normal distribution $\mathcal{N}(0, \sigma^2 I)$.

Linear regression is usually solved by maximizing loglikelihood function (2).

$$\mathcal{L}_{lr}(\beta, \sigma^2) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \|y - X\beta\|^2_2 \tag{2}$$

where $\| \cdot \|_2$ is an $\ell_2$ norm.

The estimators of model coefficients, variance and variance-covariance matrix are shown in (3).

$$\hat{\beta} = (X^T X)^{-1} X^T y$$
$$\hat{\sigma}^2 = \frac{1}{n}(y - X\hat{\beta})^T (y - X\hat{\beta})$$
$$\hat{V} = \hat{\sigma}^2 (X^T X)^{-1} \tag{3}$$

Without loss of generality, the method applied in linear regression can also be extended to polynomial regression.

2.2 Weighted Linear Regression

The weighted linear regression is similar to linear regression, except it assumes all the off-diagonal entries of the correlation matrix of the residuals are 0. By means of minimizing the corresponding SSE cost function in (4), the estimators of the model coefficients, variance and variance-covariance matrix are shown in (5).

$$SSE_{wlr}(\beta_w) = \left\| W^{1/2}(y - X\beta_w) \right\|^2_2 \tag{4}$$

where $W$ is a diagonal matrix of weights.

$$\hat{\beta}_w = (X^T WX)^{-1} X^T W y$$
$$\hat{\sigma}_w^2 = \frac{1}{n}(y - X\hat{\beta}_w)^T W (y - X\hat{\beta}_w)$$
$$\hat{V}(\hat{\beta}_w) = \hat{\sigma}_w^2 (X^T WX)^{-1} \tag{5}$$

2.3 Box-Cox Regression

Box-Cox regression model is a more generalized version of linear regression which places an additional power transformation on the response variable of the linear regression model, as shown in (6).

$$y^c = X\beta + \varepsilon \tag{6}$$

where $c$ in $y^c$ is the power parameter.

Normally, a set $C$ of power parameters is applied to the response variable. In this case, for every $c \in C$, the one maximizes the profile loglikelihood (7) is chosen as the best power parameter.

$$\mathcal{L}_{bc}(c, \beta_c, \sigma_c^2) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log \sigma_c^2$$
$$-\frac{1}{2\sigma_c^2} (y^c - X\beta_c)^T (y^c - X\beta_c) - (c - 1)^T \log y \tag{7}$$

The estimator of the model coefficients, variance and variance-covariance matrix for Box-Cox regression are

$$\hat{\beta}_c = (X^T X)^{-1} X^T y^c$$
$$\hat{\sigma}_c^2 = \frac{1}{n}(y^c - X\hat{\beta}_c)^T (y^c - X\hat{\beta}_c)$$
$$\hat{V}(\hat{\beta}_c) = \hat{\sigma}_c^2 (X^T X)^{-1} \tag{8}$$

2.4 Ridge Regression

The difference that distinguishes ridge regression and linear models is that an $\ell_2$ penalty term is added to the SSE of linear regression. The modified SSE cost function is:

$$SSE_{ridge}(\lambda, \beta) = \|y - X\beta\|^2_2 + n\lambda \|\beta\|^2_2 \tag{9}$$

where $\lambda$ is a non-negative tuning parameter used to control the penalty magnitude. For any $\lambda \geq 0$, (9) can be analytically minimized, yielding the estimator of $\beta$ as

$$\hat{\beta}_\lambda = (X^T X + \lambda I)^{-1} X^T y$$
$$\hat{\sigma}_\lambda^2 = \frac{1}{n}(y - X\hat{\beta}_\lambda)^T (y - X\hat{\beta}_\lambda)$$
$$\hat{V}(\hat{\beta}_\lambda) = \hat{\sigma}_\lambda^2 (X^T X + \lambda I)^{-1} X^T X (X^T X + \lambda I)^{-1} \tag{10}$$

3 Methodology

The main goal is to find approaches that are able to overcome the memory barrier issue of closed-form solutions and make them as widely applicable as the iterative methods in big data. In pursuit of this goal, the array of sufficient statistics (SS) is formally defined. Subsequently, multiple learning algorithms based on SS could significantly reduce the size of memory needed in the computation of the estimators for linear regression, weighted linear regression, Box-Cox regression and ridge regression are presented in this section.

3.1 Sufficient Statistics Array

SS array is an array of sufficient statistics used to calculate the estimators of the models and the loglikelihood function (or SSE cost function) without a second visit to the dataset. It’s inspired by the computation-wise row-independent of the equivalent forms of (3) of linear regression [Zhang and Yang, 2017a; Zhang and Yang, 2017b].

Noting from the equivalent forms of (3) in (11), $\sum_{i=1}^n x_i x_i^T$ is computation-wise row-independent, i.e., for any two observations $x_{i1}$ and $x_{i2}$ calculating the summation of $x_{i1} x_{i1}^T$ doesn’t depend on $x_{i2}$. Likewise, $\sum_{i=1}^n x_i y_i$ and $\sum_{i=1}^n y_i^2$ are computation-wise row-independent as well.

$$\hat{\beta} = \left( \sum_{i=1}^n x_i x_i^T \right)^{-1} \left( \sum_{i=1}^n x_i y_i \right)$$
$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n y_i^2$$
$$-\frac{1}{n} \left[ \left( \sum_{i=1}^n x_i y_i \right)^T \left( \sum_{i=1}^n x_i x_i^T \right)^{-1} \left( \sum_{i=1}^n x_i y_i \right) \right] \tag{11}$$

$$\hat{V}(\hat{\beta}) = \hat{\sigma}^2 \left( \sum_{i=1}^n x_i x_i^T \right)^{-1}$$
Inspired by this thought, the array of SS is formally defined as follows.

**Definition 1.** Sufficient statistics (SS) array is an array of sufficient statistics that computed at per row level or per mini batch level from the dataset and can be used to compute the estimators of the model coefficients \( \hat{\beta} \), the variance \( \hat{\sigma}^2 \), the variance-covariance matrices \( \hat{V}(\hat{\beta}) \) and the loglikelihood (or SSE cost function) without revisiting the dataset.

### 3.2 Linear Regression

Based on (2) and (3), \( S_{lr} \) is presented as an array of SS for linear regression.

\[
S_{lr} = (s_{yy}, s_{xy}, S_{xx}) = (\sum_{i=1}^{n} s_{yy,i}, \sum_{i=1}^{n} s_{xy,i}, S_{xx,i}) \quad (12)
\]

where \( s_{yy,i} = y_i^2 \) is a scalar, \( s_{xy,i} = x_i y_i \) is a \( p \times 1 \) vector, and \( S_{xx,i} = w_i x_i x_i^\top \) is a \( p \times p \) matrix.

By (12), we obtain the following

\[
\hat{\beta} = S_{xx}^{-1} s_{xy} \\
\hat{\sigma}^2 = \frac{1}{n} (s_{yy} - s_{xy}^\top S_{xx}^{-1} s_{xy}) \\
\hat{V}(\hat{\beta}) = \hat{\sigma}^2 S_{xx}^{-1} \quad (13)
\]

**Theorem 1.** \( S_{lr} \) is an array of SS for linear regression to derive \( \hat{\beta}, \hat{\sigma}^2, \hat{V}(\hat{\beta}) \) and \( L_{lr}(\beta, \sigma^2) \).

**Proof.** From (12), the loglikelihood can be expressed as a function of \( S_{lr} \).

\[
L_{lr}(\beta, \sigma^2) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (s_{yy} - 2s_{xy}^\top \beta + \beta^\top S_{xx} \beta) \quad (14)
\]

which only depends on SS for linear regression.

To accelerate the computation, row-by-row calculation could be optimized by batch-by-batch computation, i.e. \( \sum_{i=1}^{n} y_i^2, \sum_{i=1}^{n} x_i y_i, \sum_{i=1}^{n} x_i^2 x_i \), could be written in the form of batch:

\[
s_{yy} = \sum_{k=1}^{m} s_{yy}^{(k)} = \sum_{k=1}^{m} y_k^2 \\
s_{xy} = \sum_{k=1}^{m} s_{xy}^{(k)} = \sum_{k=1}^{m} X_k^\top y_k \\
S_{xx} = \sum_{k=1}^{m} S_{xx}^{(k)} = \sum_{k=1}^{m} X_k^\top X_k \quad (15)
\]

where \( m \) denotes the total number of batches, \( s_{yy}^{(k)}, s_{xy}^{(k)} \) and \( S_{xx}^{(k)} \) denotes SS array in batch \( k \). \( y_k \) is a \( m_k \times 1 \) vector, \( X_k \) is a \( m_k \times m_k \) array and \( m_k \) is the batch size for batch \( k \). The multiple learning approach for linear regression algorithm by batch is shown in **Algorithm 1**.

### 3.3 Weighted Linear Regression

Weighted linear regression uses weights to adjust the importance of different observations. Therefore, the SS array \( S_{wlr} \) for weighted linear regression is slightly different.

\[
S_{wlr} = (s_{wyy}, s_{wxy}, S_{wxx}) = (\sum_{i=1}^{n} s_{wyy,i}, \sum_{i=1}^{n} s_{wxy,i}, S_{wxx,i}) \quad (16)
\]

where \( s_{wyy,i} = w_i y_i^2 \) is scalar, \( s_{wxy,i} = x_i w_i y_i \) is a \( p \times 1 \) vector, and \( S_{wxx,i} = w_i x_i x_i^\top \) is a \( p \times p \) matrix.

The estimators are re-expressed as follows:

\[
\hat{\beta}_w = S_{wxx}^{-1} s_{wxy} \\
\hat{\sigma}_w^2 = \frac{1}{n} (s_{wyy} - s_{wxy}^\top S_{wxx}^{-1} s_{wxy}) \\
\hat{V}(\hat{\beta}_w) = \hat{\sigma}_w^2 S_{wxx}^{-1} \quad (17)
\]

**Theorem 2.** \( S_{wlr} \) is an array of SS for weighted linear regression to derive the estimators of \( \hat{\beta}_w, \hat{\sigma}_w^2, \hat{V}(\hat{\beta}_w) \) and \( SSE_{wlr}(\beta_w) \).

**Proof.** From (16), (4) can be expressed as a function of the SS array

\[
SSE_{wlr}(\beta_w) = s_{wyy} - 2s_{wxy}^\top \beta_w + \beta_w^\top S_{wxx} \beta_w \quad (18)
\]

which only depends on SS for weighted linear regression.

Similar to multiple learning approach for linear regression algorithm, calculating SS batch by batch is also feasible.

\[
s_{wyy} = \sum_{k=1}^{m} s_{wyy}^{(k)} = \sum_{k=1}^{m} y_k^2 W_k y_k \\
s_{wxy} = \sum_{k=1}^{m} s_{wxy}^{(k)} = \sum_{k=1}^{m} X_k^\top W_k y_k \\
S_{wxx} = \sum_{k=1}^{m} S_{wxx}^{(k)} = \sum_{k=1}^{m} X_k^\top W_k X_k \quad (19)
\]

where \( W_k \) is a \( m_k \times m_k \) diagonal weight matrix in batch \( k \).

The multiple learning approach for weighted linear regression is shown in **Algorithm 2**.
3.4 Box-Cox Regression

Box-Cox regression requires a power transformation on the response variable. Commonly, a set $C$ of power parameters are applied. And the $c$ maximizes the (7) is picked as the best parameter. As the profile loglikelihood is required for parameter picking, $(c - 1)^T \log y$ is necessarily needed.

The arrays of SS for Box-Cox regression is shown in (20). For every $c \in C$,

$$S_{c,bc} = (s_{c,yy}, s_{logy}, s_{c,xy}, S_{xx})$$

$$= \left( \sum_{i=1}^{n} s_{c,yy}, i, \sum_{i=1}^{n} s_{logy}, i, \sum_{i=1}^{n} s_{c,xy}, i, S_{xx} \right)$$

(20)

where $s_{c,yy,i} = y_i^{2c}$ and $s_{logy,i} = \log y_i$ are scalars, $s_{c,xy,i} = x_i y_i^c$ is a $p \times 1$ vector and $S_{xx,i} = x_i x_i^T$ is a $p \times p$ matrix. Notably, $S_{xx}$ is sharable to all models.

Thus, for every $c \in C$,

$$\hat{\beta}_c = S_{xx}^{-1}s_{c,xy}$$

$$\hat{\sigma}_c^2 = \frac{1}{n} \left( s_{c,yy} - s_{c,xy}^T S_{xx}^{-1} s_{c,xy} \right)$$

$$\hat{V}(\hat{\beta}_c) = \hat{\sigma}_c^2 S_{xx}^{-1}$$

(21)

Theorem 3. For any $c \in C$, the corresponding $S_{c,bc}$ is an array of SS for Box-Cox regression, which can be used to compute $\hat{\beta}_c$, $\hat{\sigma}_c^2$, $\hat{V}(\hat{\beta}_c)$ and $\mathcal{L}_{bc}(c, \beta_c, \sigma_c^2)$.

Proof. By (21), (7) becomes

$$\mathcal{L}_{bc}(\beta_c, \sigma_c^2) = -\frac{n}{2} \log(2\pi \sigma_c^2)$$

$$- \frac{1}{2\sigma_c^2} \left( s_{c,yy} - 2s_{c,xy}^T \beta_c + \beta_c^T S_{xx} \beta_c \right) + (c - 1) s_{logy}$$

(22)

which only depends on SS for Box-Cox linear regression. $\square$

Batched version of SS for $c \in C$ is shown in (23).

Algorithm 2 Weighted Linear Regression with Sufficient Statistics

**Input:** batch by batch of the entire dataset

**Output:** $\hat{\beta}, \hat{\sigma}^2$ and $\hat{V}(\hat{\beta})$

1: $s_{uyy} = 0, s_{uxy} = 0, S_{xxx} = 0$
2: for $k \leftarrow 1$ to $m$ do
3: Compute $s_{uyy}^{(k)}, s_{uxy}^{(k)}, S_{xxx}^{(k)}$ based on (19)
4: $s_{uyy} = s_{uyy}^{(k)}, s_{uxy} = s_{uxy}^{(k)}, S_{xxx} = S_{xxx}^{(k)}$
5: end for
6: if $S_{xxx}$ is singular then
7: Compute $S_{xxx}^{-1}$ using generalized inverse
8: else
9: Compute $S_{xxx}^{-1}$
10: end if
11: Compute $\hat{\beta}_c, \hat{\sigma}_c^2$ and $\hat{V}(\hat{\beta}_c)$ based on (17)
12: return $\hat{\beta}_c, \hat{\sigma}_c^2$ and $\hat{V}(\hat{\beta}_c)$

Algorithm 3 Box-Cox Regression with Sufficient Statistics by Mini-Batch

**Input:** batch by batch of the entire dataset

**Output:** $\hat{\beta}_{best}, \hat{\sigma}_{best}^2$ and $\hat{V}(\hat{\beta}_{best})$

1: $S_{xx} = 0$
2: for $c \in C$ do
3: $s_{c,yy} = 0, s_{c,xy} = 0$
4: end for
5: for $k \leftarrow 1$ to $m$ do
6: Compute $S_{xx}$ based on (23)
7: $S_{xx} = S_{xxx}^{(k)}$
8: for $c \in C$ do
9: Compute $s_{c,yy}^{(k)}, s_{c,xy}^{(k)}$ based on (23)
10: $s_{c,yy} = s_{c,yy}^{(k)}, s_{c,xy} = s_{c,xy}^{(k)}$
11: end for
12: end for
13: if $S_{xx}$ is singular then
14: Compute $S_{xx}^{-1}$ using generalized inverse
15: else
16: Compute $S_{xx}^{-1}$
17: end if
18: for $c \in C$ do
19: Compute $\hat{\beta}_c, \hat{\sigma}_c^2$ and $\hat{V}(\hat{\beta}_c)$ based on (21)
20: Compute $\mathcal{L}_{bc}(c, \beta_c, \sigma_c^2)$ based on (22)
21: end for
22: return $\hat{\beta}_{best}, \hat{\sigma}_{best}^2$ and $\hat{V}(\hat{\beta}_{best})$ based on $\mathcal{L}_{bc}$

$$s_{c,yy} = \sum_{k=1}^{m} s_{c,yy}^{(k)} = \sum_{k=1}^{m} (y_k^c)^T y_k$$

$$s_{c,xy} = \sum_{k=1}^{m} s_{c,xy}^{(k)} = \sum_{k=1}^{m} X_k^T y_k^c$$

(23)

$$S_{xx} = \sum_{k=1}^{m} S_{xx}^{(k)} = \sum_{k=1}^{m} X_k^T X_k$$

where $y_k^c$ is a $m_k \times 1$ vector in batch $k$.

The SS-based Box-Cox regression algorithm is presented in Algorithm 3.

3.5 Ridge Regression

Although ridge regression requires a set $D$ of ridge parameters, the SS array is re-usable to all ridge parameters and could be borrowed directly from linear regression.

Let $S_{ridge} = S_{lr}$, for every $\lambda \in D$, the corresponding estimators $\beta_\lambda, \sigma_\lambda^2, \hat{V}(\beta_\lambda)$ and the SSE cost function are:

$$\hat{\beta}_\lambda = (S_{xx} + \lambda I)^{-1}s_{xy}$$

$$\hat{\sigma}_\lambda^2 = \frac{1}{n} (s_{yy} - s_{xy}^T (S_{xx} + \lambda I)^{-1}s_{xy})$$

$$\hat{V}(\beta_\lambda) = \hat{\sigma}_\lambda^2 (S_{xx} + \lambda I)^{-1} S_{xx} (S_{xx} + \lambda I)$$

(24)

$$SSE_{ridge}(\lambda, \beta_\lambda) = \| y - X\beta_\lambda \|_2^2 + n\lambda \| \beta_\lambda \|_2^2$$

(25)

The best $\lambda$ is selected by the ridge trace method.
Theorem 4. $S_{\text{ridge}}$ is the SS array for ridge regression.

Proof. From (24), (9) could be expressed as

$$SSE_{\text{ridge}}(\lambda, \beta_\lambda) = s_{yy} - 2s_{xy}^\top \beta_\lambda + \beta_\lambda^\top S_{xx} \beta_\lambda + n \lambda \beta_\lambda^\top \beta_\lambda$$

(26)

which only depends on SS for ridge regression. □

The batched version for SS is also identical to that of linear regression.

The multiple learning approach for ridge regression algorithms is presented in Algorithm 4.

4 Experiments

To evaluate the proposed multiple learning algorithms, extensive experiments were carried out on a computer cluster containing four computers. All algorithms are implemented on Spark.

We designed two experiments, one for time performance and the other for prediction quality, to compare the results between the multiple learning algorithms and the traditional ones on Spark.

Time Performance Comparison The first experiment is to evaluate the time used for training different models. It compared the time performance of multiple learning approaches with fixed batch size and the traditional approaches. We simulated three datasets with different sizes: 1GB, 10GB and 100GB. The datasets are generated with $\varepsilon \sim \mathcal{N}(0, I)$ and each row has 100 features for this testing. For Box-Cox regression, three similar datasets are sampled but all labels are ensured to be non-negative.

Table 1 compared time performance for different models. In Table 1, (i) Spark represents the traditional approaches implemented by Apache Spark, however, Box-Cox regression has no official implementation on Spark, thus we provided an alternative version; (ii) SS 1 (SS 128) means the multiple learning approaches with batch size fixed to 1 (128); (iii) As Box-Cox regression accept multiple power parameters, we chose $C = [1 \text{ to } 1.9]$ by an interval of 0.1. Likewise, $D = [0 \text{ to } 0.9]$ by an interval of 0.1 is used for ridge regression.

| Model       | Time Used (s) |
|-------------|---------------|
|             | 1GB           | 10GB          | 100GB         |
| LR          | Spark         | 41.86         | 338.27        | 3266.16       |
|             | SS 1          | 19.59         | 154.16        | 1505.64       |
|             | SS 128        | 15.67         | 126.33        | 1267.96       |
| Weighted LR | Spark         | 42.23         | 339.54        | 3267.37       |
|             | SS 1          | 19.76         | 155.47        | 1528.75       |
|             | SS 128        | 16.73         | 133.35        | 1289.54       |
| Box-Cox     | $C = [1]$     | 42.63         | 341.31        | 3264.33       |
|             | SS 1          | 19.16         | 156.41        | 1532.00       |
|             | SS 128        | 15.10         | 122.49        | 1200.49       |
| Box-Cox     | $C = [1 \text{ to } 1.9]$ | 431.29        | 3429.34       | 3370.51       |
|             | SS 1          | 19.87         | 160.13        | 1674.62       |
|             | SS 128        | 16.10         | 122.21        | 1206.17       |
| Ridge       | $D = [0.1]$   | 41.58         | 328.48        | 3276.10       |
|             | SS 1          | 19.41         | 147.22        | 1502.96       |
|             | SS 128        | 19.41         | 147.22        | 1502.96       |
| Ridge       | $D = [1 \text{ to } 1.9]$ | 423.63        | 3342.58       | 32688.28      |
|             | SS 1          | 20.56         | 152.47        | 1620.46       |
|             | SS 128        | 15.10         | 125.63        | 1230.45       |

Table 1: Time Performance Comparison

| Model    | MSE           |
|----------|---------------|
|          | 1GB           | 10GB          | 100GB         |
| Spark LR | 1014481.73    | 992885.63     | 994789.20     |
| SS LR    | 1011518.85    | 990132.98     | 994213.58     |

Table 2: MSE comparison

Based on the results from Table 1, the training time of our methods is much shorter than that of the traditional ones on Spark. According to Table 1, the performance of traditional Box-Cox and ridge regression are affected dramatically by the number of power parameters and ridge parameters. The slow performance is because Box-Cox and ridge regression needs datasets reloading and retraining for each power and ridge parameter. In contrast, our approaches are only slightly affected by the number of power parameters and ridge parameters as all models are calculated simultaneously via multiple SS arrays. From this, we could conclude that, if a large scale and hard-to-fit dataset is given, our approaches can outperform the traditional ones by avoiding revisiting the dataset. This advantage is vital if the convergence requirements are hard to meet.

Prediction Quality Comparison To experimentally support that our algorithms are as accurate as OLS algorithms with one pass through the datasets, we compared our algorithms with the traditional ones. In this experiment, we used 1GB, 10GB and 100GB as training sets and an additional 0.2GB, 2GB and 20GB data for testing. Mean square error (MSE) is used as the metric. Due to page limitation, we did not elaborate the MSE for all models, but Table 2 is sufficient to show that our approaches can produce closed-form solutions.
5 Conclusion

In this work, multiple learning approaches for regression are proposed for big data. With only one pass through the dataset, SS array is computed to derive the closed-form solutions for linear regression, weighted linear regression, Box-Cox regression and ridge regression. Theoretically and experimentally, it’s proven that multiple learning is capable of overcoming the memory barrier issue. Furthermore, multiple SS arrays could be applied to obtain multiple models at one time. Results showed our approaches are extremely efficient when calculating multiple models as opposed to the traditional methods. However, multiple learning is limited to closed-form solutions. In the future, we hope to conduct more experiments on real large scale datasets and extend multiple learning to the field of non closed-form solutions.

References

[Altay and Satman, 2005] Erdiç Altay and M Hakan Satman. Stock market forecasting: artificial neural network and linear regression comparison in an emerging market. Journal of Financial Management & Analysis, 18(2):18, 2005.

[Avriel, 2003] Mordecai Avriel. Nonlinear programming: analysis and methods. Courier Corporation, 2003.

[Barata and Hussein, 2012] João Carlos Alves Barata and Mahir Saleh Hussein. The moore–penrose pseudoinverse: A tutorial review of the theory. Brazilian Journal of Physics, 42(1-2):146–165, 2012.

[Ben-Israel and Greville, 2003] Adi Ben-Israel and Thomas NE Greville. Generalized inverses: theory and applications, volume 15. Springer Science & Business Media, 2003.

[Box and Cox, 1964] George EP Box and David R Cox. An analysis of transformations. Journal of the Royal Statistical Society: Series B (Methodological), 26(2):211–243, 1964.

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pages 785–794. ACM, 2016.

[Davidof, 2010] William C. Davidof. Variable metric method for minimization*.

[Dennis and Moré, 1977] John E Dennis, Jr and Jorge J Moré. Quasi-newton methods, motivation and theory. SIAM review, 19(1):46–89, 1977.

[DeSarbo and Cron, 1988] Wayne S DeSarbo and William L Cron. A maximum likelihood methodology for clusterwise linear regression. Journal of classification, 5(2):249–282, 1988.

[Dresden, 2009] Arnold Dresden. The fourteenth western meeting of the american mathematical society. Bull. Amer. Math. Soc., 26(9):385–396, 06 1920.

[He et al., 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[Hoefl and Kennard, 1970] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1):55–67, 1970.

[Holland and Welsch, 1977] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. Communications in Statistics-theory and Methods, 6(9):813–827, 1977.

[Kenney and Keeping, 1962] John F Kenney and ES Keeping. Linear regression and correlation. Mathematics of statistics, 1:252–285, 1962.

[Kiwiel, 2001] Krzysztof K Kiwiel. Convergence and efficiency of subgradient methods for quasiconvex minimization. Mathematical programming, 90(1):1–25, 2001.

[Malouf, 2002] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In proceedings of the 6th conference on Natural language learning. Volume 20, pages 1–7. Association for Computational Linguistics, 2002.

[Marquardt, 1970] Donald W Marquardt. Generalized inverses, ridge regression, biased linear estimation, and nonlinear estimation. Technometrics, 12(3):591–612, 1970.

[Moler, 1986] Cleve Moler. Matrix computation on distributed memory multiprocessors. Hypercube Multiprocessors, 86(181-195):31, 1986.

[Myers and Myers, 1990] Raymond H Myers and Raymond H Myers. Classical and modern regression with applications, volume 2. Duxbury Press Belmont, CA, 1990.

[Naseem et al., 2010] Imran Naseem, Roberto Togneri, and Mohammed Bennamoun. Linear regression for face recognition. IEEE transactions on pattern analysis and machine intelligence, 32(11):2106–2112, 2010.

[Ngheip and Al, 2001] Nguyen Ngheip and Cripps Al. Predicting housing value: A comparison of multiple regression analysis and artificial neural networks. Journal of real estate research, 22(3):313–336, 2001.

[Pedregosa et al., 2011] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. Journal of machine learning research, 12(Oct):2825–2830, 2011.

[Ruder, 2016] Sebastian Ruder. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747, 2016.

[Wedderburn, 1974] Robert WM Wedderburn. Quasi-likelihood functions, generalized linear models, and the gauss—newton method. Biometrika, 61(3):439–447, 1974.
[Zaharia et al., 2010] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. HotCloud, 10(10-10):95, 2010.

[Zhang and Yang, 2017a] Tonglin Zhang and Baijian Yang. Box–cox transformation in big data. Technometrics, 59(2):189–201, 2017.

[Zhang and Yang, 2017b] Tonglin Zhang and Baijian Yang. An exact approach to ridge regression for big data. Computational Statistics, 32(3):909–928, 2017.