A Selective Review on Statistical Techniques for Big Data

Yaqiong Yao and HaiYing Wang

Abstract To meet the big data challenges, many new statistical tools have been developed in recent years. In this review, we summarize some of these approaches to give an overview of the current state of the development. We will focus on the case that the number of observations is much larger than the dimension of the unknown parameters, although we will mention some investigations related to the high-dimensional data. We will discuss methods using subsamples as well as methods processing the whole data piece-by-piece.

Key words: Divide-and-Conquer; IBOSS; Optimal Subsampling; Randomized Numerical Linear Algebra; Online Updating

1 Introduction

As the collection and storage of data becoming much cheaper than before, volumes of available data are increasing exponentially and big data problems attract a wide range of attentions from scientists [13]. In many disciplines, a lot of data with extraordinary sizes emerge and need more advanced technologies and approaches to analyze them, because traditional methods may fail due to large data volumes. For big data, size is not the only concern. The difficulty of analyzing big data can be evaluated in three aspects: volume, velocity and variety. Here, volume is the size related to both the dimension and the number of observations; velocity is the interaction speed with the data; and variety means various data structures [20].

In this review, we mainly discuss the case that the number of observations far exceeds the data dimension, and consider two challenges caused by big data. The

Yaqiong Yao
University of Connecticut, Storrs, Connecticut, USA, e-mail: yaqiong.yao@uconn.edu

HaiYing Wang
University of Connecticut, Storrs, Connecticut, USA, e-mail: haiying.wang@uconn.edu

one is that analyzing the entire dataset is time-consuming and the other is that the data is too large to be held in the computer's random access memory (RAM). To deal with these two problems, a bunch of statistical methods have been developed: some methods target at one of the challenges and some methods are useful to meet both challenges.

To speed up calculation for the first challenge, one may project the massive dataset to a lower dimensional space using some randomized transforms. This procedure is called random projection, and existing methods often rely on the Hadamard transform or the Johnson-Lindenstrauss transform [10, 26].

Another intuitive solution is to select a subset of the full data to analyze. This approach can be implemented through a random subsampling procedure or a deterministic selection method. For random subsampling, nonuniform subsampling probabilities are often used for spotting more informative data points [e.g. 11]. Algorithmic leveraging is an example of this procedure, which uses statistical leverage scores to define subsampling probabilities for linear regression models [24]. Another example is the local case control subsampling which is designed for logistic regression with imbalanced data [14]. Optimal subsampling is a recently developed technique that derives the optimal subsampling probabilities by minimizing the asymptotic mean squared error of the resulting subsample estimator. This method typically needs to be implemented in an adaptive way because the optimal subsampling probabilities contain unknown parameters [38]. The information-based optimal subdata selection is a novel deterministic selection method designed for linear regression models [37]. This methods has the advantage that the relevant information contained in the resulting subsample is not restricted by the subsample size.

When the data volume is so large that the whole data cannot be analyzed in the available RAM, one solution is to process the data piece-by-piece. The divide-and-conquer method is a typical example of this approach. With this method, one divides the entire dataset into small blocks, analyzes data in each blocks, and then aggregates results from all blocks to form a final estimator [e.g. 22]. Stochastic gradient descent is another example of processing the data piece-by-piece. This method reads the observations one-by-one or batch-by-batch; and it updates the estimator step-by-step, so there is no need to store the data that have been used.

Besides the approaches that we are going to discuss in this chapter, there are a bunch of other methods focusing on a particular model when responses could be correlated, such as resampling-based stochastic approximation method [21] and multi-resolution approximation method [18] for Gaussian processes, online asynchronous decentralized leverage score sampling for vector autoregressive model [41].

The rest of the paper is organized as follows. Section 2 discusses the randomized numerical linear algebra including methods based on random projection and random subsampling. Section 3 presents the information-based optimal subdata selection methods. Section 4 is devoted to informative subsampling methods including the optimal subsampling methods and local case control subsampling methods. Section 5 presents divide-and-conquer methods, online updating methods, and stochastic gradient descent methods. Section 6 gives brief summary and discussions.

2 Randomized Numerical Linear Algebra

Suppose that $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ are n observations with $\boldsymbol{x}_i \in \mathbb{R}^d$ being the covariate and y_i being the response. Assume that they follow a linear regression model with the following form

$$y_i = \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta} + \varepsilon_i, \quad i = 1, ..., n, \tag{1}$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$ is the unknown regression coefficient and $\{\varepsilon_i\}_{i=1}^n$ are uncorrelated error terms with $\mathbb{E}(\varepsilon_i) = 0$ and $\mathbb{V}(\varepsilon_i) = \sigma^2$. This model can also be written as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{2}$$

where $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)^{\mathrm{T}} \in \mathbb{R}^{n\times d}$ is the covariate matrix or design matrix, $\boldsymbol{y} = (y_1, y_2, ..., y_n)^{\mathrm{T}}$ is the n dimensional vector of responses, and $\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, ..., \varepsilon_n)^{\mathrm{T}}$ is the n dimensional vector of model errors.

The ordinary least-squares (OLS) estimator is commonly used to estimate $\boldsymbol{\beta}$, and it has a form of

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|^2 = \boldsymbol{X}^{+}\boldsymbol{y}, \tag{3}$$

where $\|\cdot\|$ represents the Euclidean norm and $\boldsymbol{X}^{+}$ is the Moore-Penrose inverse of $\boldsymbol{X}$. If $\boldsymbol{X}$ is a full-rank matrix, $\hat{\boldsymbol{\beta}}$ has an expression of

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = \left(\sum_{i=1}^n \boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\right)^{-1} \sum_{i=1}^n \boldsymbol{x}_i y_i \tag{4}$$

Under the setting of $n > d$, both (3) and (4) can be computed in $O(nd^2)$ time. However, for extremely large n and potentially large d, obtaining the OLS estimator is not trivial. One reason is that the $O(nd^2)$ computational time may not be affordable, and another reason is that the size of the dataset may exceed the capacity of available RAM. In the following, we discuss two methods proposed to fast approximate the OLS estimator: one is based on random projection after the Hadamard transform and the other is based on nonuniform subsampling according to leverage scores of the design matrix. For other investigations on randomized numerical linear algebra, readers can refer to [26, 27] and the references therein.

2.1 Random Projection

Random projection is a widely used method, which reduces the dimension of a matrix by mapping it to a comparatively low dimensional space with a relatively small error [17, 26]. A random projection method for solving the least-squares problem

was introduced in [10], which combines random projection (and uniform sampling) with the randomized Hadamard transform (also known as the Walsh–Hadamard transform).

Before presenting this method, we need to introduce the Hadamard transform, which is defined recursively through Hadamard matrix. Suppose that $\boldsymbol{H}_n$ represents the $n \times n$ Hadamard matrix in which $n = 2^a$ for some positive integer a. When $n = 2$ ($a = 1$), define

$$\boldsymbol{H}_2 = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix}.$$

The Hadamard matrix $\boldsymbol{H}_n$ is defined as

$$\boldsymbol{H}_n = \boldsymbol{H}_2 \otimes \boldsymbol{H}_{n/2},$$

where $\otimes$ is the kronecker product. An computational advantage of the Hadamard transform is that it takes $O(n \log_2 n)$ time to obtain $\boldsymbol{H}_n \boldsymbol{x}$ for any $\boldsymbol{x} \in \mathbb{R}^n$. For example, this can be achieved by using Algorithm 1. However, an disadvantage of the Hadamard transform is that n has to be a power of 2.

Algorithm 1 Fast Walsh-Hadamard Transform (FWHT)

Input: $\boldsymbol{x} \in \mathbb{R}^n$ with $n = 2^a$
Output: $\eta = \boldsymbol{H}_n \boldsymbol{x}$ = FWHT($\boldsymbol{x}$)

1: **if** $n = 2$ **then**
2: $\eta = \begin{pmatrix} x_1 + x_2 \\ x_1 - x_2 \end{pmatrix}$ where $\boldsymbol{x} = (x_1, x_2)^{\mathrm{T}}$,
3: **else**
4: partition $\boldsymbol{x}$ into $\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}^1 \\ \boldsymbol{x}^2 \end{pmatrix}$, where $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are of the same dimension, and calculate $\eta^1 \leftarrow$ FWHT($\boldsymbol{x}^1$), $\eta^2 \leftarrow$ FWHT($\boldsymbol{x}^2$), and $\eta = \begin{pmatrix} \eta^1 + \eta^2 \\ \eta^1 - \eta^2 \end{pmatrix}$,
5: **end if**

The basic idea of the proposed algorithm in [10] is to first average the information of observations by using the Hadamard transform on both the response vector and the design matrix, and then randomly project the transformed data into a lower dimensional space or uniformly draw data points from the transformed data. The final estimator, say $\tilde{\boldsymbol{\beta}}^{\mathrm{S}}$, is calculated based on the projected data or the selected subsample from the transformed data. The algorithm based on random projection is described in Algorithm 2.

Algorithm 2 Random Projection after Randomized Hadamard Transform

Input: $\boldsymbol{X}$, $\boldsymbol{y}$, r, q

Output: $\tilde{\boldsymbol{\beta}}^{\mathrm{S}}$

1: Let $\S \in \mathbb{R}^{r\times n}$ be a sparse projection matrix such that for $i = 1,2,...,r$ and $j = 1,2,...,n$,

$$\S_{ij} = \begin{cases} +\sqrt{\frac{1}{rq}}, & \text{with probability } \frac{q}{2}, \\ -\sqrt{\frac{1}{rq}}, & \text{with probability } \frac{q}{2}, \\ 0, & \text{with probability } 1-q, \end{cases} \qquad \text{independently.}$$

2: Let $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ be a diagonal matrix whose diagonal elements are $+1$ or -1 with equal probability.

3: The estimator $\tilde{\boldsymbol{\beta}}^{\mathrm{S}}$ is obtained as

$$\tilde{\boldsymbol{\beta}}^{\mathrm{S}} = (\S \boldsymbol{H}_n \boldsymbol{R} \boldsymbol{X})^{+} \S \boldsymbol{H}_n \boldsymbol{R} \boldsymbol{y}.$$

Using Algorithm 1, the term $\boldsymbol{H}_n\boldsymbol{R}\boldsymbol{X}$, in Algorithm 2 can be calculated in $O(nd\log_2 n)$ time, which is the major computational cost in the algorithm if § is sparse enough. Theorem 3 of [10] provides formulas for determining the values of r and q based on a desired level of relative approximation error.

In [10], the authors also considered the case that § is a subsampling matrix defined in this way: for $i = 1,2,...,r$, randomly choose j from $\{1,2,...,n\}$, and set $\S_{ij} = 1$ and $\S_{ij'} = 0$ for $j' \neq j$. This is just to take a subsample from the transformed data using uniform subsampling with replacement. There is a computational benefit of using this approach. We actually only keep r rows of $\boldsymbol{H}_n\boldsymbol{R}\boldsymbol{X}$ after multiplying the sampling matrix §, the time complexity of $\S\boldsymbol{H}_n\boldsymbol{R}\boldsymbol{X}$ is $O(nd\log_2 r)$ according to [4]. The authors also proved, in Theorem 2 of [10], that the value of r should be

$$r = \max\left(48^2 d\ln(40nd)\ln\{100^2 d\ln(40nd)\}, 40d\ln(40nd)/\alpha\right),$$

in order to achieve the $(1+\alpha)$ relative error approximation with high probability, where $\alpha \in (0,1)$. The overall time complexity of the proposed algorithm is $O(nd\ln d)$ if $d \leq n \leq \exp(d)$. The full data needs to be read in one time to conduct the randomized Hadamard transform. Based on the approach in [10], a least-squares solver `BLENDENPIK` is developed in [5].

Note that $\boldsymbol{H}_n\boldsymbol{H}_n = n\boldsymbol{I}$ and $\boldsymbol{R}\boldsymbol{R} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. Thus the randomized Hadamard transform does not change the full data OLS estimator, because for OLS estimator based on the transformed data

$$\hat{\boldsymbol{\beta}}_{\boldsymbol{H}_n} = \{(\boldsymbol{H}_n\boldsymbol{R}\boldsymbol{X})^{\mathrm{T}}\boldsymbol{H}_n\boldsymbol{R}\boldsymbol{X}\}^{-1}(\boldsymbol{H}_n\boldsymbol{R}\boldsymbol{X})^{\mathrm{T}}\boldsymbol{H}_n\boldsymbol{R}\boldsymbol{y} = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} = \hat{\boldsymbol{\beta}}.$$

However, the randomized Hadamard transform make the data points more similar so a uniform subsampling will not miss any very informative data points. This also indicates that if the data have light-tailed distributions, the randomized Hadamard transform may not be very effective. For example, if $(\boldsymbol{x}_i, y_i)$ are independent and

identically distributed (i.i.d) with a multivariate normal distribution, then the randomized Hadamard transformed data follows the same distribution.

2.2 Nonuniform Random Sampling

Instead of transforming the data to be more uniform and then using uniform sampling, another idea is to use nonuniform subsampling and assign higher probabilities to more informative data points. We obtain the least-squares or weighted least-squares estimator based on the sample drawn according to those nonuniform subsampling probabilities.

A general approach of nonuniform subsampling for the overconstrained linear regression problem was proposed in [11], which only needs to process the full data by one pass. Leverage scores are commonly used to construct nonuniform subsampling probabilities, and this kind of algorithms is summarized in [24] and is named as algorithmic leveraging.

Suppose that $\boldsymbol{X}$ is full rank with singular value decomposition (SVD),

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}, \tag{5}$$

where $\boldsymbol{U}$ is a $n \times d$ orthonormal matrix, $\boldsymbol{V}$ is a $d \times d$ orthonormal matrix, and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ is a diagonal matrix with diagonal elements being the singular values. Denote each row of $\boldsymbol{U}$ as $\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n$. The leverage scores are

$$h_i = \|\boldsymbol{u}_i\|^2, \quad i = 1, 2, ..., n,$$

which are equivalent to

$$h_i = \boldsymbol{x}_i^{\mathrm{T}}(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{x}_i, \quad i = 1, 2, ..., n,$$

and they satisfies that $\sum_{i=1}^{n} h_i = d$.

The algorithmic leveraging is to use normalized leverage scores, $d^{-1}h_i$, $i = 1, ..., n$, to define a subsampling distribution. Algorithm 3 describes how to obtain the algorithmic leveraging estimator $\tilde{\boldsymbol{\beta}}^{\mathrm{AL}}$.

The authors of [24] investigated the properties of $\tilde{\boldsymbol{\beta}}^{\mathrm{AL}}$ and proposed the shrinkage leveraging (SLEV) method which uses

$$\left\{ \pi_i = \rho\frac{h_i}{d} + (1-\rho)\frac{1}{n} \right\}_{i=1}^{n},$$

where $\rho \in (0, 1)$ is a tuning parameter. They showed that the SLEV estimator often has a smaller variance. In addition, the asymptotic normality and unbiasedness of $\tilde{\boldsymbol{\beta}}^{\mathrm{AL}}$ has been examed in [25] under some regularity conditions.

Another issue of algorithmic leveraging is that the leverage scores need $O(nd^2)$ time to compute. To alleviate the computational burden, [9] proposed to fast approx-

Algorithm 3 Algorithmic Leveraging

Input: $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, r

Output: $\tilde{\boldsymbol{\beta}}^{\mathrm{AL}}$

1: Sample with replacement for a subsample of size r from the full data, using the sampling distribution

$$\left\{\pi_i = \frac{h_i}{d}\right\}_{i=1}^n.$$

2: Denote the selected subsample and the corresponding subsampling probabilities as $\{\boldsymbol{x}_i^*, y_i^*, \pi_i^*\}_{i=1}^r$. The estimator $\tilde{\boldsymbol{\beta}}^{\mathrm{AL}}$ is

$$\tilde{\boldsymbol{\beta}}^{\mathrm{AL}} = \left(\sum_{i=1}^r \frac{\boldsymbol{x}_i^* \boldsymbol{x}_i^{*\mathrm{T}}}{\pi_i^*}\right)^{-1} \sum_{i=1}^r \frac{\boldsymbol{x}_i^* y_i^*}{\pi_i^*}.$$

imate leverage scores h_i's by using

$$\tilde{h}_i = \|(\boldsymbol{X}(\Pi_1 \boldsymbol{X})^+ \Pi_2)_{i*}\|^2, \quad i = 1, ..., n,$$

where $\Pi_1 =\in \mathbb{R}^{r_1 \times n}$ is the subsampled randomized Hadamard transform (e.g., §$\boldsymbol{H}_n\boldsymbol{R}$ in Algorithm 2) and $\Pi_2 \in \mathbb{R}^{r_1 \times r_2}$ is the Johnson-Lindenstrauss transform (JLT) [2, 3]. To obtain the JLT, each entry of Π_2 is generated independently as

$$\Pi_{2(i,j)} = \begin{cases} +\sqrt{3/r_2} & \text{with probability } \frac{1}{6} \\ -\sqrt{3/r_2} & \text{with probability } \frac{1}{6} \\ 0 & \text{with probability } \frac{2}{3} \end{cases}.$$

The values of r_1 and r_2 are discussed in Lemme 6 and Lemma 4 of [9], respectively. This algorithm runs in $O(nd \ln n)$ time if $d \leq n$ and $n = o(e^d)$.

In stead of solving a subsample OLS problem, [7] suggested to use the following estimator to estimate the true parameter

$$\tilde{\boldsymbol{\beta}}^{\mathrm{NS}} = \left(\sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}\right)^{-1} \sum_{i=1}^r \frac{\boldsymbol{x}_i^* y_i^*}{\pi_i^*},$$

for the situation with measurement constraints. This is a scenario that all $\boldsymbol{x}_i$'s are available but the number of responses that can be measured is limited, and the goal is to sample $\boldsymbol{x}_i$'s and then measure the corresponding values of y_i's to estimate $\boldsymbol{\beta}$. This is a typical problem of interest in the field of survey sampling and design of experiments in statistics. The estimator $\tilde{\boldsymbol{\beta}}^{\mathrm{NS}}$ does not improve the computational efficiency compared with the full data OLS. However, there is an explicit formula for the mean squared error (MSE) of $\tilde{\boldsymbol{\beta}}^{\mathrm{NS}}$, and the authors obtained the optimal subsampling probabilities under the A-optimality criteria in optimal design.

Linear regression with measurement constraints is further discussed in [40]. Let r be the number of y_i's can be measured. They proposed to first obtain a computationally tractable relaxed A-optimal design,

$$\boldsymbol{\zeta}^0 = \arg\min_{\boldsymbol{\zeta}=\{\zeta_i\}_{i=1}^n} \text{tr}\left\{\left(\sum_{i=1}^n \zeta_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_i\right)^{-1}\right\}, \quad \text{subject to} \quad 0 \le \zeta_i \text{ and } \sum_{i=1}^n \zeta_i \le r.$$

Additionally for Poisson subsampling, it is required that $\max(\zeta_i) \le 1$. With $\boldsymbol{\zeta}^0 = \{\zeta_i^0\}_{i=1}^n$, they assign the subsampling probabilities as

$$\left\{\pi_i^{(1)} = \frac{\zeta_i^0 \boldsymbol{x}_i^{\mathrm{T}}(\sum_{j=1}^n \zeta_j^0 \boldsymbol{x}_j^{\mathrm{T}}\boldsymbol{x}_j)^{-1}\boldsymbol{x}_i}{d}\right\}_{i=1}^n \quad \text{for subsampling with replacement; and}$$

$$\left\{\pi_i^{(2)} = \frac{\zeta_i^0}{r}\right\}_{i=1}^n \quad \text{for Poisson subsampling.}$$

3 Information-Based Optimal Subdata Selection

The methods discussed in previous sections are based on random subsampling or random projection. The asymptotic variances of the resulting estimators are typically at the orders of the inverse subsample sizes. This means that if the subsample size r does not go to infinity, then the subsample estimator does not converge to the true parameter, no matter how fast the full data sample size n goes to infinity. In other words, the subsample estimator is not consistent if r does not go to infinite. The information-based optimal subdata selection (IBOSS) aims to solve this issue.

The IBOSS method was proposed in [37] for the linear regression model (1) or (2). Here, the linear regression model contains an intercept parameter, and we emphasize this fact by writing $\boldsymbol{x}_i = (1, x_{i,1}, ..., x_{i,d-1})^{\mathrm{T}}$ for $i = 1, ..., n$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, ..., \beta_{d-1})^{\mathrm{T}}$.

Unlike the random sampling approaches we have discussed, the IBOSS deterministically select data points according to some optimality criterion on the information matrix. Suppose that $\mathscr{D} = \{\boldsymbol{x}_i^*, y_i^*\}_{i=1}^r$ is a deterministic subset of the full dataset and the selection rule depends on $\boldsymbol{X} = (\boldsymbol{x}_1^{\mathrm{T}}, ..., \boldsymbol{x}_n^{\mathrm{T}})^{\mathrm{T}}$ only. Since selection rule does not depend on $\boldsymbol{y} = (y_1, y_2, ..., y_n)^{\mathrm{T}}$, based on the subsample, the least-squares estimator

$$\tilde{\boldsymbol{\beta}} = \left\{\sum_{i=1}^r \boldsymbol{x}_i^* \boldsymbol{x}_i^{*\mathrm{T}}\right\}^{-1} \sum_{i=1}^r \boldsymbol{x}_i^* y_i^* \tag{6}$$

is still the best linear unbiased estimator of $\boldsymbol{\beta}$. The information matrix of the subsample corresponding to the least-squares estimator for $\boldsymbol{\beta}$ is

$$I(\mathscr{D}) = \frac{1}{\sigma^2}\sum_{i=1}^{r} \boldsymbol{x}_i^* \boldsymbol{x}_i^{*\mathrm{T}}.$$

The IBOSS aims to select a subsample that, under some optimality criterion, maximizes $I(\mathscr{D})$, which is equivalent to minimize the covariance matrix of $\tilde{\boldsymbol{\beta}}$. Under the D-optimality criterion, one needs to find the subsample that maximizes $|\sum_{i=1}^{n} \boldsymbol{x}_i^* \boldsymbol{x}_i^{*\mathrm{T}}|$. Since there are $\binom{n}{r}$ different subsamples, the exact solution is computational infeasible due to the combinatorial time complexity. The authors derived an upper bound of $|\sum_{i=1}^{n} \boldsymbol{x}_i^* \boldsymbol{x}_i^{*\mathrm{T}}|$, which indicates that the D-optimal subsample are related to the extremes of the covariates. Based on this observation, they proposed to select $k = [\frac{r}{2(d-1)}]$ observations corresponding to the smallest and largest values of each covariate variable, where $[\cdot]$ means rounding to the nearest integer. The procedure of IBOSS is summarized in Algorithm 4.

Algorithm 4 IBOSS

Input: $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, $k = [\frac{r}{2(d-1)}]$

Output: $\tilde{\boldsymbol{\beta}}^{\mathrm{IBOSS}}$

Initializing: $\mathscr{D} \leftarrow \varnothing$, $\mathscr{D}^c \leftarrow \{\boldsymbol{x}_i, y_i\}_{i=1}^n$,

1: **for** $j \in \{1,2,3,...,d-1\}$ **do**
2: with a partition-based selection algorithm, choose the observations in $\mathscr{D}^c$ with the k smallest values of $x_{i,j}$ and the k largest values of $x_{i,j}$; record these $2k$ observations as $\mathscr{D}_j$;
3: update $\mathscr{D}^c \leftarrow \mathscr{D}^c \setminus \mathscr{D}_j$ and $\mathscr{D} \leftarrow \mathscr{D} \cup \mathscr{D}_j$;
4: **end for**
5: Calculate $\tilde{\boldsymbol{\beta}}^{\mathrm{IBOSS}}$ defined in (6) using the observations in $\mathscr{D}$.

The IBOSS procedure in Algorithm 4 has a time complexity of $O(nd)$, which is a linear time in terms of the full data and is faster than the algorithms described in Section 2. The authors investigated the variance of the resulting estimator $\tilde{\boldsymbol{\beta}}^{\mathrm{IBOSS}}$ in various settings. One key conclusion is that the variance for a slope estimator may converge to zero at a rate related to both r and n. Theorem 5 of [37] shows that the variance of a slope estimator satisfies

$$\mathbb{V}(\tilde{\boldsymbol{\beta}}_j^{\mathrm{IBOSS}}|\boldsymbol{X}) = O_P\left(\frac{d-1}{r\{x_{(n-k+1),j} - x_{(k),j}\}^2}\right), \quad j = 1,2,...,d-1,$$

where $x_{(i),j}$ is the i-th order statistics of $x_{1,j},...,x_{n,j}$. This result indicates that even if r is fixed, the variance of a slope estimator can still go to 0 as n increases if the support of the covariate distribution is not bounded. For example, if the covariate follows a t distribution with degrees of freedom ν, then the corresponding slope estimator has a variance of order $\mathbb{V}(\tilde{\boldsymbol{\beta}}_j^{\mathrm{IBOSS}}|\boldsymbol{X}) = O_P(r^{-1}n^{-2/\nu})$. In addition, every covariate should be read into memory in one time to select the subdata. Since the data are stored in hard disk by row, the IBOSS fails when full data volume exceeds the capacity of available RAM. To solve this, the IBOSS was combined with

the divide-and-conquer approach in [34]. Another benefit of this investigation is to utilize the distributed and parallel computing facilities.

4 Informative Subsampling

In sections 2 and 3, the subsampling approaches do not depend on the responses, and our discussion has focused on linear regression models. In this section, we will introduce some informative subsampling methods that are applicable to non-linear models. By informative subsampling, we mean that the subsampling depends on the responses.

4.1 Optimal Subsampling

Optimal subsampling is an informative subsampling approach that aims to maximize the estimation efficiency. The basic strategy is to find subsampling probabilities that minimize the asymptotic variance of the subsample estimator.

The optimal subsampling method under the A-optimality criterion (OSMAC) was first introduced for logistic regression in [38], where the authors derived subsampling probabilities that minimize the asymptotic MSE of the subsample approximation error.

Consider a logistic regression model,

$$P(y_i = 1|\boldsymbol{x}_i) = p(\boldsymbol{x}_i, \boldsymbol{\beta}) = \frac{\exp(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})}{1+\exp(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})}, \quad i = 1, 2, ..., n, \tag{7}$$

where $y_i \in \{0,1\}$ is the response, $\boldsymbol{x}_i \in \mathbb{R}^d$ is the covariate, and $\boldsymbol{\beta}$ is the unknown regression coefficient. Let $\{\boldsymbol{x}_i^*, y_i^*, \pi_i^*\}_{i=1}^r$ be a subsample taken according to subsampling probabilities $\{\pi_i\}_{i=1}^n$ such that $\sum_{i=1}^n \pi_i = 1$. A general subsample estimator is

$$\tilde{\boldsymbol{\beta}}^{\mathrm{gen}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{r} \frac{y_i^* \boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i^* - \log\left(1 + e^{\boldsymbol{\beta}^{\mathrm{T}} \boldsymbol{x}_i^*}\right)}{\pi_i^*}, \tag{8}$$

which aims to approximate the full data maximum likelihood estimator (MLE), denoted as $\hat{\boldsymbol{\beta}}_{\mathrm{MLE}}$. The authors of [38] derived the following optimal subsampling probabilities that minimize the asymptotic MSE of the approximation error $\tilde{\boldsymbol{\beta}}^{\mathrm{gen}} - \hat{\boldsymbol{\beta}}_{\mathrm{MLE}}$,

$$\pi_i = \frac{|y_i - p(\boldsymbol{x}_i, \hat{\boldsymbol{\beta}}_{\mathrm{MLE}})| \|\boldsymbol{M}_x^{-1} \boldsymbol{x}_i\|}{\sum_{j=1}^n |y_j - p(\boldsymbol{x}_j, \hat{\boldsymbol{\beta}}_{\mathrm{MLE}})| \|\boldsymbol{M}_x^{-1} \boldsymbol{x}_j\|}, \quad i = 1, ..., n, \tag{9}$$

where $\boldsymbol{M}_x = \frac{1}{n}\sum_{i=1}^n p(\boldsymbol{x}_i, \hat{\boldsymbol{\beta}}_{\mathrm{MLE}})\{1 - p(\boldsymbol{x}_i, \hat{\boldsymbol{\beta}}_{\mathrm{MLE}})\}\boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}}$. Since (9) contains $\hat{\boldsymbol{\beta}}_{\mathrm{MLE}}$, the full data MLE, the author proposed an adaptive algorithm stated in Algorithm 5.

Note that, to reduce the computational burden, $\boldsymbol{M}_x$ can be approximated by pilot sample. Using this way, the approximated optimal subsampling probabilities can be computed by going through the full data once. After sampling the index of the subample from 1 to n under the approximated optimal subsampling probabilities, the subsample is obtained by reading in the full data in one pass.

Algorithm 5 Two-Stage Adaptive Subsampling

Input: $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$, r_0, r

Output: $\tilde{\boldsymbol{\beta}}^{\text{OS}}$

1: **Pilot sampling:** Sample with replacement for a subsample of size r_0, $\{\boldsymbol{x}_i^{*0}, y_i^{*0}, \pi_i^{*0}\}_{i=1}^{r_0}$, using uniform sampling $\{\pi_i^0 = n^{-1}\}_{i=1}^n$ or case control sampling $\{\pi_i^0 = (2n_0)^{-y_i+1}(2n_1)^{-y_i}\}_{i=1}^n$, where n_0 and n_1 are the number of 0's and 1's, respectively, in the responses. Obtain the pilot estimator $\tilde{\boldsymbol{\beta}}^{*0}$ and substitute $\hat{\boldsymbol{\beta}}_{\text{MLE}}$ in (9) with $\tilde{\boldsymbol{\beta}}^{*0}$ to calculate the approximated optimal probabilities $\tilde{\pi}_i$.

2: **Second stage sampling:** Sample with replacement based on $\{\tilde{\pi}_i\}_{i=1}^n$ for a subsample of size r, $\{\boldsymbol{x}_i^*, y_i^*, \tilde{\pi}_i^*\}_{i=1}^r$.

3: **Estimation:** The final estimator $\tilde{\boldsymbol{\beta}}^{\text{OS}}$ is obtained by combing the two state samples

$$\tilde{\boldsymbol{\beta}}^{\text{OS}} = \arg\max_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{r_0} \frac{y_i^{*0}\boldsymbol{\beta}^{\text{T}}\boldsymbol{x}_i^{*0} - \log\left(1+e^{\boldsymbol{\beta}^{\text{T}}\boldsymbol{x}_i^{*0}}\right)}{\pi_i^{*0}} + \sum_{i=1}^{r} \frac{y_i^{*}\boldsymbol{\beta}^{\text{T}}\boldsymbol{x}_i^{*} - \log\left(1+e^{\boldsymbol{\beta}^{\text{T}}\boldsymbol{x}_i^{*}}\right)}{\tilde{\pi}_i^{*}} \right\}. \quad (10)$$

The authors of [38] also proved the consistency of the resultant estimator to $\hat{\boldsymbol{\beta}}_{\text{MLE}}$ given the full data and derived its asymptotic distribution. This method was improved in [35] in terms of the estimation efficiency by using an unweighted target function. Specially, instead of using (10), the author suggested to obtain

$$\tilde{\boldsymbol{\beta}}^{\text{uw}} = \arg\max_{\boldsymbol{\beta}} \sum_{i=1}^{r} \left\{ y_i^{*}\boldsymbol{\beta}^{\text{T}}\boldsymbol{x}_i^{*} - \log\left(1+e^{\boldsymbol{\beta}^{\text{T}}\boldsymbol{x}_i^{*}}\right) \right\}, \quad (11)$$

correct the bias of $\tilde{\boldsymbol{\beta}}^{\text{uw}}$ to have $\tilde{\boldsymbol{\beta}}^{\text{bs}} = \tilde{\boldsymbol{\beta}}^{\text{uw}} + \tilde{\boldsymbol{\beta}}^{*0}$, and then aggregate $\tilde{\boldsymbol{\beta}}^{\text{bs}}$ and $\tilde{\boldsymbol{\beta}}^{*0}$ instead of combining the two stage subsamples. Here the bias correction procedure is similar to that proposed for the local case control subsampling method [14] to be discussed in the next section. The author of [35] also investigated the Poisson subsampling procedure, and showed that it has a higher estimation efficiency in addition to its computational benefits.

Although the OSMAC was proposed in the context of logistic regression, it is applicable to other statistical models. It has been generalized in [42] to softmax regression, in which the response y_i has $B+1$ possible values, $0, 1, 2, ..., B$, with the following probabilities

$$\begin{aligned} P(y_i = 0|\boldsymbol{x}_i) &= p_i(0, \boldsymbol{\beta}) = \frac{1}{1+\sum_{j=1}^B \exp(\boldsymbol{x}_i^{\text{T}}\boldsymbol{\beta}_j)}, \quad \text{and} \\ P(y_i = b|\boldsymbol{x}_i) &= p_i(b, \boldsymbol{\beta}) = \frac{\exp(\boldsymbol{x}_i^{\text{T}}\boldsymbol{\beta}_b)}{1+\sum_{j=1}^B \exp(\boldsymbol{x}_i^{\text{T}}\boldsymbol{\beta}_j)}, \quad m = 1, 2, ..., B. \end{aligned} \quad (12)$$

Here $\boldsymbol{\beta}_b \in \mathbb{R}^d$ is the regression coefficient for the b-th category and we set $\boldsymbol{\beta}_0 = \mathbf{0}$ for model identifiability. The whole unknown parameter vector is $\boldsymbol{\beta} = \{\boldsymbol{\beta}_1^{\mathrm{T}}, \boldsymbol{\beta}_2^{\mathrm{T}}, ..., \boldsymbol{\beta}_B^{\mathrm{T}}\}^{\mathrm{T}}$. The optimal subsampling probabilities under the A-optimality criterion used to draw subsamples for approximating the full data MLE $\hat{\boldsymbol{\beta}}_{\text{MLE}}$ are

$$\pi_i = \frac{\|\boldsymbol{M}_S^{-1}\{\boldsymbol{s}_i(\hat{\boldsymbol{\beta}}_{\text{MLE}}) \otimes \boldsymbol{x}_i\}\|}{\sum_{j=1}^n \|\boldsymbol{M}_S^{-1}\{\boldsymbol{s}_j(\hat{\boldsymbol{\beta}}_{\text{MLE}}) \otimes \boldsymbol{x}_j\}\|}, \quad i = 1, ..., n,$$

where $\boldsymbol{M}_S = n^{-1}\sum_{i=1}^n \Upsilon_i(\hat{\boldsymbol{\beta}}_{\text{MLE}}) \otimes (\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}})$; $\Upsilon_i(\boldsymbol{\beta})$ is a $B \times B$ matrix whose b-th diagonal element is $\Upsilon_{i,(b,b)}(\boldsymbol{\beta}) = p_i(b,\boldsymbol{\beta}) - p_i^2(b,\boldsymbol{\beta})$ and b_1b_2-th off-diagonal element is $\Upsilon_{i,(b_1,b_2)}(\boldsymbol{\beta}) = -p_i(b_1,\boldsymbol{\beta})p_i(b_2,\boldsymbol{\beta})$; and $s_i(\boldsymbol{\beta}) \in \mathbb{R}^B$ with b-th element being $\boldsymbol{s}_{i,b}(\boldsymbol{\beta}) = I(y_i = b) - p_i(b,\boldsymbol{\beta})$.

In [1], the OSMAC was generalized to include generalized linear models (GLMs) with the following form

$$f(\boldsymbol{y}_i, |\boldsymbol{x}_i, \boldsymbol{\beta}) = h(y_i)\exp\left[y_i g(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}) - c\{g(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta})\}\right], \tag{13}$$

where $h(\cdot)$, $g(\cdot)$ and $c(\cdot)$ are known functions. The optimal subsampling probabilities under A-optimality for approximating the full data MLE $\hat{\boldsymbol{\beta}}_{\text{MLE}}$ are

$$\pi_i = \frac{|y_i - \dot{c}\{g(\boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\}|\|\boldsymbol{M}_G^{-1}\dot{g}(\boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\boldsymbol{x}_i\|}{\sum_{j=1}^n |y_j - \dot{c}\{g(\boldsymbol{x}_j^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\}|\|\boldsymbol{M}_G^{-1}\dot{g}(\boldsymbol{x}_j^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\boldsymbol{x}_j\|}, \quad i = 1, ..., n,$$

where $\dot{c}(\cdot)$ and $\dot{g}(\cdot)$ are the first-order derivatives of $c(\cdot)$ and $g(\cdot)$, respectively; and

$$\boldsymbol{M}_G = \frac{1}{n}\sum_{i=1}^n \{\ddot{g}(\boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}[\dot{c}\{g(\boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\} - y_i] + \ddot{c}\{g(\boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\}\dot{g}^2(\boldsymbol{x}_i^{\mathrm{T}}\hat{\boldsymbol{\beta}}_{\text{MLE}})\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}\},$$

with $\ddot{c}(\cdot)$ and $\ddot{g}(\cdot)$ being the second-order derivatives of $c(\cdot)$ and $g(\cdot)$, respectively.

The OSMAC was extended to quantile regression in [36], which assume that the τ-th quantile ($0 < \tau < 1$) of the response y_i at the give value $\boldsymbol{x}_i$ satisfies

$$q_\tau(y_i|\boldsymbol{x}_i) = \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}.$$

The regression coefficient $\boldsymbol{\beta}$ is estimated through minimizing

$$Q_N(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^n (y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_i)\{\tau - I(y_i < \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}_i)\}, \tag{14}$$

and the L-optimal subsampling probabilities used to draw subsamples for approximating the full data estimator are

$$\pi_i = \frac{|\tau - I(y_i - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta} < 0)|\|\boldsymbol{x}_i\|}{\sum_{j=1}^n |\tau - I(y_j - \boldsymbol{x}_j^{\mathrm{T}}\boldsymbol{\beta} < 0)|\|\boldsymbol{x}_j\|}, \quad i = 1, ..., n. \tag{15}$$

To obtain the A-optimal subsampling probabilities, one just replace $\|\boldsymbol{x}_i\|$ in (15) with $\|\boldsymbol{M}_Q^{-1}\boldsymbol{x}_j\|$, where $\boldsymbol{M}_Q = n^{-1}\sum_{i=1}^n f_{\boldsymbol{x}_i}(0)\boldsymbol{x}\boldsymbol{x}_i^{\mathrm{T}}$, and $f_{\boldsymbol{x}_i}(0)$ is the density function of $y_i - \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{\beta}$ evaluated at 0 for a given $\boldsymbol{x}_i$. For quantile regression, the authors recommended the L-optimality over the A-optimality because $f_{\boldsymbol{x}_i}(0)$'s in $\boldsymbol{M}_Q$ are typically infeasible to obtain. In addition, the L-optimal subsampling probabilities takes $O(nd)$ time to calculate while the A-optimal subsampling probabilities takes $O(nd^2)$ time to calculate even if $\boldsymbol{M}_Q$ is available. To perform statistical inference without estimating $f_{\boldsymbol{x}_i}(0)$'s, the authors proposed an iterative subsampling procedure based on the L-optimal probabilities.

Using the similar idea of OSMAC to approximate full data maximum quasi-likelihood estimator $\hat{\boldsymbol{\beta}}_{\mathrm{QLE}}$ by Poisson subsampling was discussed in [43]. A distributed sampling system based on the divide-and-conquer method was also introduced.

4.2 Local Case Control Subsampling

Local case control (LCC) sampling was proposed by [14] for logistic regression model (7) with imbalanced datasets. Unlike the case control sampling in which the subsampling probabilities only depend on the responses $\{y_i\}_{i=1}^n$, the LCC subsampling probabilities depend on both the responses and the covariates. Specifically, the LCC subsampling probabilities for estimating the parameter $\boldsymbol{\beta}$ in the logistic regression model (7) are

$$\pi_i^{\mathrm{LCC}} = |y_i - p(\boldsymbol{x}_i, \tilde{\boldsymbol{\beta}}^p)|, \quad i = 1, ..., n, \tag{16}$$

where $\tilde{\boldsymbol{\beta}}^p$ is a pilot estimator. The LCC subsampling is based on Poisson sampling. A detail algorithm for estimating the regression parameter $\boldsymbol{\beta}$ is presented in Algorithm 6. Given pilot estimator, the subsample is obtained by reading the full data once because we can calculate π_i^{LCC} based on i-th observation, and determine whether to include i-th observation in the sample or not immediately after knowing π_i^{LCC}.

From Algorithm 6, the actual subsample size is random, and with a consistent pilot estimator, the expected subsample size is asymptotically $n\mathbb{E}|y - p(\boldsymbol{x}, \boldsymbol{\beta})| = 2n[p(\boldsymbol{x}, \boldsymbol{\beta})\{1 - p(\boldsymbol{x}, \boldsymbol{\beta})\}] \leq n/2$. Although this expected subsample size is at the same order of the full data sample size n, it can be much smaller than half of the full sample size for very imbalanced data. The authors derived the asymptotic distribution of $\tilde{\boldsymbol{\beta}}^{\mathrm{LCC}}$ unconditionally on the full data. The asymptotic variance of $\tilde{\boldsymbol{\beta}}^{\mathrm{LCC}}$ is twice as large as than that of the full data MLE, if the logistic regression model is correct.

The idea of LCC sampling is extended to the softmax regression model (12) in [15], and the method is named as local uncertainty sampling (LUS). Given a pilot estimator $\tilde{\boldsymbol{\beta}}^p$, the authors proposed the following subsampling probabilities

Algorithm 6 Local Case Control Subsampling

Input: $\{\boldsymbol{x}_i, y\}_{i=1}^n$, $\tilde{\boldsymbol{\beta}}^p$ (a pilot estimator)
Output: $\tilde{\boldsymbol{\beta}}^{\text{LCC}}$
Initializing: $\mathscr{D} \leftarrow \varnothing$
for i in $\{1,2,...,n\}$ **do**
  generate $u_i \sim \text{Uniform}(0,1)$ and calculate $\pi_i^{\text{LCC}} = |y_i - p(\boldsymbol{x}_i, \tilde{\boldsymbol{\beta}}^p)|$
  if $u_i < \pi_i^{\text{LCC}}$ **then**
    $\mathscr{D} \leftarrow \mathscr{D} \cup \{(\boldsymbol{x}_i, y_i)\}$
  end if
end for
Estimation: Denote the obtained subsample as $\mathscr{D} = \{\boldsymbol{x}_i^*, y_i^*\}_{i=1}^r$. Calculate $\tilde{\boldsymbol{\beta}}^{\text{uw}}$ according to (11) and the final LCC estimator is $\tilde{\boldsymbol{\beta}}^{\text{LCC}} = \tilde{\boldsymbol{\beta}}^{\text{uw}} + \tilde{\boldsymbol{\beta}}^p$.

$$\pi(\boldsymbol{x}_i, y_i) = \begin{cases} \frac{1-q_i}{\theta - \max(q_i, 0.5\theta)} & \text{if } p_i(y_i, \tilde{\boldsymbol{\beta}}^p) = q_i \\ \min(1, 2q_i/\theta) & \text{Otherwise} \end{cases},$$

where $\theta \geq 1$ is a pre-specified number so that the expected subsample size is no more than n/θ; and $q_i = \max\{0.5, p_i(0, \tilde{\boldsymbol{\beta}}^p), p_i(1, \tilde{\boldsymbol{\beta}}^p), p_i(2, \tilde{\boldsymbol{\beta}}^p), .., p_i(B, \tilde{\boldsymbol{\beta}}^p)\}$. Denote the obtained subsample as $\mathscr{D} = \{\boldsymbol{x}_i^*, y_i^*\}_{i=1}^r$. The authors of [15] derived the conditional distribution of y_i^* given $\boldsymbol{x}_i^*$ as follows,

$$\begin{aligned} P(y_i^* = 0 | \boldsymbol{x}_i^*) &= \frac{1}{1 + \sum_{j=1}^B \exp\left\{\boldsymbol{\beta}_j^{\mathrm{T}} \boldsymbol{x}_i^* + \log \frac{\pi(\boldsymbol{x}_i^*, j)}{\pi(\boldsymbol{x}_i^*, 0)}\right\}}, \\ P(y_i^* = b | \boldsymbol{x}_i^*) &= \frac{\exp\left\{\boldsymbol{\beta}_b^{\mathrm{T}} \boldsymbol{x}_i^* + \log \frac{\pi(\boldsymbol{x}_i^*, b)}{\pi(\boldsymbol{x}_i^*, 0)}\right\}}{1 + \sum_{j=1}^B \exp\left\{\boldsymbol{\beta}_j^{\mathrm{T}} \boldsymbol{x}_i^* + \log \frac{\pi(\boldsymbol{x}_i^*, j)}{\pi(\boldsymbol{x}_i^*, 0)}\right\}}, \quad b = 1, 2, ..., B. \end{aligned} \tag{17}$$

Note that Poisson subsampling is used so $\{\boldsymbol{x}_i^*, y_i^*\}_{i=1}^r$ are i.i.d conditional on $\tilde{\boldsymbol{\beta}}^p$. Thus, for a given $\tilde{\boldsymbol{\beta}}^p$, (17) can be used to construct the likelihood function for the subsample, and the final estimator $\tilde{\boldsymbol{\beta}}^{\text{LUC}}$ is the MLE based on the sampled data.

5 Divide-and-Conquer and Updating Methods

This section introduces the divide-and-conquer method, which partitions the full data into smaller pieces, performs calculations on them separately, and then combines these calculation results to obtain a final estimator. Most updating methods also process data piece-by-piece, but in a sequential manner, so we will discuss several updating methods in this section as well.

5.1 Divide-and-Conquer Methods

Unlike the methods we have discussed in previous sections that use part of the original full data or transformed full data to perform the final analysis, the divide-and-conquer approach provides another scheme to deal with massive data. This method partitions the whole dataset into K pieces, processes these K pieces separately to obtain relevant subdata statistics, and then aggregates these subdata statistics to obtain the final estimator. Note that although divide-and-conquer method can take advantage of distributed and parallel computing facility, it may not save computing time if we have a single computer. Furthermore, there is no general approach to do the aggregation. One way is to use the simple average of the subdata estimators as the final estimator, but this may not have the highest estimation efficient. In the following, we will discuss how to aggregate subdata estimators for estimation equation [22] and how to determine the sparsity pattern for high dimensional case [8].

A divide-and-conquer method was investigated by [22] in terms of estimation equation. Let the independent full dataset be $\{\boldsymbol{z}_i\}_{i=1}^n$ which satisfies that $\sum_{i=1}^n \mathbb{E}\{\psi(\boldsymbol{z}_i, \boldsymbol{\beta}_t)\} = 0$ for some smooth function ψ, where $\boldsymbol{\beta}_t$ is the true parameter. The estimating equation estimator $\hat{\boldsymbol{\beta}}^{\mathrm{EE}}$ is the solution to

$$\sum_{i=1}^{n} \psi(\boldsymbol{z}_i, \boldsymbol{\beta}) = 0. \tag{18}$$

Here the model setup is quite general and it includes regression models if we let $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$. If the full data volume is too large to be loaded in one machine, [22] proposed the aggregated estimating equation (AEE) estimator as presented in Algorithm 7. The divide-and-conquer method has an obvious benefit on memory efficiency because the full data is processed block by block, and we only record $\{\boldsymbol{C}_{\tilde{\boldsymbol{\beta}}_k}, \tilde{\boldsymbol{\beta}}_k\}$ for each block.

It was proved that $\tilde{\boldsymbol{\beta}}^{\mathrm{AEE}}$ is consistent to the original full data estimation equation estimator $\hat{\boldsymbol{\beta}}^{\mathrm{EE}}$ under some regularity conditions [22].

The divide-and-conquer approach is applied to the generalized linear model (13) with high dimensional data in [8]. A penalty term is added to the log-likelihood in this setting to ensure sparsity, and the estimator is named as split-and-conquer estimator. The full data penalized estimator $\hat{\boldsymbol{\beta}}^{\mathrm{PL}}$ is defined as

$$\hat{\boldsymbol{\beta}}^{\mathrm{PL}} = \arg\max_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \frac{\log f(y_i|\boldsymbol{x}_i, \boldsymbol{\beta})}{n} - \lambda(\boldsymbol{\beta}, b) \right\},$$

where $\lambda(\cdot)$ is the penalty function to ensure sparsity (some components of $\hat{\boldsymbol{\beta}}^{\mathrm{PL}}$ are 0), and b is the tunning parameter. The full dataset is partitioned into K blocks, and within each block, a coefficient estimator $\tilde{\boldsymbol{\beta}}_k^{\mathrm{PL}}$ is obtained for $k = 1, 2, ..., K$. Since $\tilde{\boldsymbol{\beta}}_k^{\mathrm{PL}}$'s are calculated from different data blocks, their sparsity patterns are typically different, and this complicates the aggregation step. The authors of [8] proposed

Algorithm 7 Divide-and-Conquer for estimation equation

Input: $\{\boldsymbol{z}_i\}_{i=1}^n$

Output: $\tilde{\boldsymbol{\beta}}^{\text{AEE}}$

1: Partition the full dataset into K blocks, $\{\boldsymbol{z}_i^k\}_{i=1}^{n_k}$, where $n_1,...,n_K$ ($\sum_{k=1}^K n_k = n$) are the number of observations in each block, respectively, and $\boldsymbol{z}_i^k$ is the i-th observation in the k-th block.

2: **for** $k \in \{1,2,...K\}$ **do**

3: obtain the k-th block estimator $\tilde{\boldsymbol{\beta}}_k$ by solving

$$\sum_{i=1}^{n_k} \psi(\boldsymbol{z}_i^k, \boldsymbol{\beta}) = 0;$$

4: calculate

$$\boldsymbol{C}_{\tilde{\boldsymbol{\beta}}_k} = -\sum_{i=1}^{n_k} \dot{\psi}(\boldsymbol{z}_i^k, \tilde{\boldsymbol{\beta}}_k),$$

where $\dot{\psi}(\cdot)$ is the gradient of $\psi(\cdot)$ with respect to $\boldsymbol{\beta}$.

5: Record $\{\boldsymbol{C}_{\tilde{\boldsymbol{\beta}}_k}, \tilde{\boldsymbol{\beta}}_k\}$.

6: **end for**

7: Obtain the aggregated estimating equation estimator as

$$\tilde{\boldsymbol{\beta}}^{\text{AEE}} = \left\{\sum_{k=1}^K \boldsymbol{C}_{\tilde{\boldsymbol{\beta}}_k}\right\}^{-1} \sum_{k=1}^K \boldsymbol{C}_{\tilde{\boldsymbol{\beta}}_k} \tilde{\boldsymbol{\beta}}_k.$$

the majority voting method to specify the sparsity pattern of the split-and-conquer estimator, denoted as $\tilde{\boldsymbol{\beta}}^{\text{SC}} = (\tilde{\beta}_1^{\text{SC}},...,\tilde{\beta}_d^{\text{SC}})^{\text{T}}$. The majority voting method sets

$$\tilde{\beta}_j^{\text{SC}} = 0 \quad \text{if } \sum_{k=1}^K I(\tilde{\beta}_{k,j}^{\text{PL}} \neq 0) \leq \omega, \quad j = 1,2,...,d$$

where $\omega \in [0,K)$ controls the number of zeros in the final estimator. If $\omega = 0$, then $\tilde{\beta}_j^{\text{SC}} = 0$ only if all $\tilde{\beta}_{k,j}^{\text{PL}}$'s are 0; if $\omega \in [K-1,K)$, then $\tilde{\beta}_j^{\text{SC}} = 0$ if any of $\tilde{\beta}_{k,j}^{\text{PL}}$ is 0. Nonzero elements of $\tilde{\boldsymbol{\beta}}^{\text{SC}}$ are obtained by aggregating the corresponding elements of $\tilde{\boldsymbol{\beta}}_k^{\text{PL}}$, $k = 1,2,...,K$. The author proved that $\tilde{\boldsymbol{\beta}}^{\text{SC}}$ and the full data estimator $\hat{\boldsymbol{\beta}}^{\text{PL}}$ are asymptotic equivalent by showing that these two estimators have the same asymptotic variances. This method reduces the computational burden from $O(n^2d)$ to $O(n^2d)/K$ given that $d \gg n$, if the LARS algorithm proposed by [12] is used for linear regression.

Some other related investigations on the divide-and-conquer approach include [6, 30, 39, 44], among others. Note that K cannot grow too fast as N goes to infinity in order to obtain a good final estimator [22]. The choice of K is discussed in [30] in the smoothing spline setting and the authors stated that the rate of K should be no faster than the sharp upper bound $O(n^{2s/(2s+1)})$ in order for the aggregated smooth spline estimator to attain the minimax optimal convergence rate, where s is

the degree of smoothness of the true regression function. The authors of [6] studied statistical inferences with the divide-and-conquer approach for high dimensional linear and generalized linear models. They obtained the order of K under which the error caused by the divide-and-conquer is negligible compared with the statistical error. In [39], the adaptive LASSO estimator for sparse Cox regression model is approximated through a three-steps divide-and-conquer algorithm. Divide-and-conquer method was introduced to kernel ridge regression in [44] in which the final estimator could achieve a minimax optimal convergence rate.

5.2 Updating Methods

The aforementioned divide-and-conquer methods assume that all blocks of data are accessible simultaneously. For streaming data, one does not have access to all the data at a time and may not be able to store all the historical data. Thus, one needs to update the estimator as new data comes in. The online updating method was introduced to deal with this situation. Stochastic gradient descent is a popular optimization method, which uses new data or sampled data to approximate the gradient of the objective function in each iterative step. This is essentially to update the estimator with new pieces of data, so we categorize it as an updating method here.

5.2.1 Online Updating Methods

Under the simple linear regression model, the online updating method for streaming data was investigated in [19], where the updating formulas for estimators of the intercept, the slope and the model error variance are derived.

An online updating method based on the divide-and-conquer technique for estimating equation (18) is proposed in [29], and a novel cumulatively updated estimating equation (CUEE) estimator is developed. Suppose that in Algorithm 7, data blocks are coming sequentially, and each block of data is accessible only once. The CUEE estimator up to block k, $\tilde{\boldsymbol{\beta}}_k^{\text{CUEE}}$, is defined as

$$\tilde{\boldsymbol{\beta}}_k^{\text{CUEE}} = \left\{ \sum_{i=1}^{k} \boldsymbol{C}_{\check{\boldsymbol{\beta}}_i} \right\}^{-1} \left\{ \sum_{i=1}^{k} \boldsymbol{C}_{\check{\boldsymbol{\beta}}_i} \check{\boldsymbol{\beta}}_i + \sum_{i=1}^{k} \boldsymbol{u}_i(\check{\boldsymbol{\beta}}_i) \right\},$$

where

$$\check{\boldsymbol{\beta}}_k = \left(\sum_{i=1}^{k-1} \boldsymbol{C}_{\check{\boldsymbol{\beta}}_i} + \boldsymbol{C}_{\tilde{\boldsymbol{\beta}}_k} \right)^{-1} \left(\sum_{i=1}^{k-1} \boldsymbol{C}_{\check{\boldsymbol{\beta}}_i} \check{\boldsymbol{\beta}}_i + \boldsymbol{C}_{\tilde{\boldsymbol{\beta}}_k} \tilde{\boldsymbol{\beta}}_k \right),$$

$\boldsymbol{u}_k(\boldsymbol{\beta}) = \sum_{i=1}^{n_k} \psi(\boldsymbol{z}_i^k, \boldsymbol{\beta})$, $\boldsymbol{C}_{\check{\boldsymbol{\beta}}_0} = \mathbf{0}_{d\times d}$ and $\check{\boldsymbol{\beta}}_0 = \mathbf{0}_d$. For the CUEE estimator, there is no need to store the raw data, and one only needs to store the following statistics

for updating: $\sum_{i=1}^{k-1} \boldsymbol{C}_{\check{\boldsymbol{\beta}}_i}$, $\sum_{i=1}^{k-1} \boldsymbol{C}_{\check{\boldsymbol{\beta}}_i}\check{\boldsymbol{\beta}}_i$, and $\sum_{i=1}^{k-1} \boldsymbol{u}_i(\check{\boldsymbol{\beta}}_i)$. When the k-th data block arrives, these statistics are updated and $\tilde{\boldsymbol{\beta}}_k^{\text{CUEE}}$ is calculated. The author proved that the CUEE estimator is consistent to $\hat{\boldsymbol{\beta}}^{\text{EE}}$ under some regularity conditions. Another interesting problem studied by [33] is how to update the estimator when new covariate variables are introduced into the model at some time point k.

A method to approximate the MLE for GLMs with streaming data is proposed in [23], in which they focus on GLMs with a dispersion parameter in addition to the mean parameter $\boldsymbol{\beta}$ as shown in model (13). Let $\boldsymbol{U}_k(\boldsymbol{\beta})$ be the score function (the gradient of the log-likelihood function) and $\boldsymbol{J}_k(\boldsymbol{\beta})$ be the negative observed information matrix (the negative Hessian matrix of the log-likelihood function), for the mean parameter based on the k-th block of data, $k = 1, ..., K$. The proposed incremental updating algorithm obtains the updated estimator $\tilde{\boldsymbol{\beta}}_k^{\text{RN}}$ when the k-th data block arrives by solving

$$\sum_{i=1}^{k-1} \boldsymbol{J}_i(\tilde{\boldsymbol{\beta}}_i^{\text{RN}})(\tilde{\boldsymbol{\beta}}_{k-1}^{\text{RN}} - \tilde{\boldsymbol{\beta}}_k^{\text{RN}}) + \boldsymbol{U}_k(\tilde{\boldsymbol{\beta}}_{k-1}^{\text{RN}}) = \boldsymbol{0}.$$

This paper also provides the updating formula for estimating the dispersion parameter enabling one to perform real-time statistical inference. The resultant estimator is consistent to the true parameter and is asymptotically normal, and these asymptotic properties are true without imposing the condition that $K = O(n_k^{\nu})$ for some $\nu < \frac{1}{3}$ and any k which is needed in [22, 29].

5.2.2 Stochastic Gradient Descent

Stochastic gradient descent (SGD) is a popular optimization technique for massive data, which recursively updates estimators, and discards the involving raw data in each step to improve the memory efficiency. Here we focus on a case of parameter estimation through the log-likelihood function which is presented as an optimization problem. Let $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be i.i.d data, and denote the log-likelihood for each data point as $\ell(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$. In this setting, the MLE $\hat{\boldsymbol{\beta}}_{\text{MLE}}$ maximizes the log-likelihood function of the full data $\sum_{i=1}^n \ell(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$, and SGD is commonly used maximization algorithm to approximate $\hat{\boldsymbol{\beta}}_{\text{MLE}}$.

A popular classic SGD algorithm is defined as

$$\tilde{\boldsymbol{\beta}}_i^{\text{sgd}} = \tilde{\boldsymbol{\beta}}_{i-1}^{\text{sgd}} + \gamma_i \boldsymbol{D}_i \dot{\ell}(y_i; \boldsymbol{x}_i, \tilde{\boldsymbol{\beta}}_{i-1}^{\text{sgd}}), \quad i = 1, 2, ..., n, \tag{19}$$

where $\dot{\ell}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta}) = \partial \log \ell(y_i; \boldsymbol{x}_i, \boldsymbol{\beta}) / \partial \boldsymbol{\beta}$ is the gradient of the log-likelihood, $\gamma_i = O(n^{-c})$ is the learning rate, $c \in (0.5, 1]$, and $\boldsymbol{D}_i$ is a positive-definite matrix which is often chosen to be the identity matrix or a diagonal matrix for computational efficiency. The learning rate is critical for the performance of an SGD algorithm. If γ_i's are too large, then the SGD procedure in (19) may not converge. To alleviate this problem, [31] proposed an implicit SGD algorithm which updates the parameter

estimate $\tilde{\boldsymbol{\beta}}_i^{\text{im}}$ in each step by solving

$$\tilde{\boldsymbol{\beta}}_i^{\text{im}} = \tilde{\boldsymbol{\beta}}_{i-1}^{\text{im}} + \gamma_i \boldsymbol{D}_i \dot{\ell}(y_i; \boldsymbol{x}_i, \tilde{\boldsymbol{\beta}}_i^{\text{im}}).$$

The implicit SGD differs from the classic SGD in that the stochastic gradient in the i-th step is evaluated at the i-th estimate instead of previous estimate in the $(t-1)$-th step. The implicit SGD converges for a larger range of the learning rate, and the resulting estimator is consistent to the true parameter and shares the same asymptotic variance as that for the classic SGD estimator under mild conditions.

Another problem of the SGD is that only one data point $(\boldsymbol{x}_i, y_i)$ is used, so the gradient may have a large variance, resulting in a slow convergence rate of the algorithm. Mini-batch SGD alleviates this issue to some extent, but the computation burden can be high for a large batch size. Several methods have been proposed to reduce the variation of the gradient. Stochastic variance reduced gradient (SVRG) proposed by [16] introduces an average gradient term, and the resulting algorithm has been proved to converge in a linear rate for a smooth and strong convex optimization target function.

The gradient variance can also be reduced by nonuniform sampling, which assigns different probabilities for different observations to be selected. This approach was discussed in [28, 45]. The nonuniform probabilities can be obtained by minimizing the variance of the gradient, which are proportional to the norms of the gradients. Calculating gradient norms for all observations in each step is computationally expensive. The authors of [45] constructed upper bounds of gradient norms that do not depend on $\boldsymbol{\beta}$ and use them to construct nonuniform sampling probabilities. Furthermore, a weighted SGD is used to ensure the unbiasedness, such as

$$\tilde{\boldsymbol{\beta}}_t^{\text{sgd}} = \tilde{\boldsymbol{\beta}}_{t-1}^{\text{sgd}} + \gamma_t \frac{\dot{\ell}(y_t; \boldsymbol{x}_t, \tilde{\boldsymbol{\beta}}_{t-1}^{\text{sgd}})}{\pi_t^{\text{IS}}}, \quad t = 1, 2, 3, ...,$$

where $(\boldsymbol{x}_t, y_t)$ is a random sample from the full data according to the sampling distribution $\{\pi_t^{\text{IS}}\}_{t=1}^n$.

Using control variate is another way to reduce the variation of the stochastic gradient [32]. One uses a vector $\tilde{\dot{\ell}}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$, which has the same expectation as $\dot{\ell}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$ but has a smaller variance, to substitute the gradient $\dot{\ell}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$. Here, $\tilde{\dot{\ell}}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$ can be constructed as

$$\tilde{\dot{\ell}}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta}) = \dot{\ell}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta}) - \boldsymbol{A}^{\text{T}} \{\kappa(y_i; \boldsymbol{x}_i, \boldsymbol{\beta}) - \kappa_e(\boldsymbol{\beta})\},$$

where $\kappa(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$ is the control variate with $\mathbb{E}\{\kappa(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})\} = \kappa_e(\boldsymbol{\beta})$ and $\boldsymbol{A}$ is a $d \times d$ matrix obtained by minimizing $\mathbb{V}\{\tilde{\dot{\ell}}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})\}$. The control variate $\kappa(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$ is expected to be highly correlated with $\dot{\ell}(y_i; \boldsymbol{x}_i, \boldsymbol{\beta})$ to achieve the desired effect of variance reduction. Different optimization problems have different control variates and a useful way is to construct the control variate is by using low-order moments. The author also discussed that $\boldsymbol{A}$ can be a diagonal matrix or even a single number to attain faster computational speed.

6 Summary and Discussion

We have selectively introduced several statistical methods aiming at reducing computational burden for massive datasets, including randomized numerical linear algebra, IBOSS, informative subsampling, divide-and-conquer and updating methods. All of these methods exhibit excellent estimation efficiency and computation efficiency. Meanwhile, they all have their limitations. Based on different datasets and situations, we need to know how to choose an appropriate method.

Even though we have so many elegant methods to deal with big data problems, there are still many research problems remaining to be solved. Most of the methods require that the model is correctly specified. However, sometimes, it is hard to build a model accurately due to the complexity and diversity of the data. How to improve the robustness of existing methods is an important topic for future investigation. In addition, real data can have various complex structures and may contain a lot of noises, measurement errors, and censoring, which increase the difficulty to perform data analysis. More advanced methods and complex models are required to fulfill this need.

References

[1] Mingyao Ai, Jun Yu, Huiming Zhang, and HaiYing Wang. Optimal subsampling algorithms for big data regressions. *Statistica Sinica, DOI: 10.5705/ss.202018.0439*, 2019.

[2] Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563, 2006.

[3] Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

[4] Nir Ailon and Edo Liberty. Fast dimension reduction using rademacher series on dual bch codes. *Discrete & Computational Geometry*, 42(4):615, 2009.

[5] H. Avron, P. Maymounkov, and S. Toledo. Blendenpik: Supercharging LAPACK's least-squares solver. *SIAM Journal on Scientific Computing*, 32:1217–1236, 2010.

[6] Heather Battey, Jianqing Fan, Han Liu, Junwei Lu, and Ziwei Zhu. Distributed testing and estimation under sparse high dimensional models. *Annals of statistics*, 46(3):1352, 2018.

[7] Siheng Chen, Rohan Varma, Aarti Singh, and Jelena Kovačcević. A statistical perspective of sampling scores for linear regression. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1556–1560. IEEE, 2016.

[8] Xueying Chen and Min-ge Xie. A split-and-conquer approach for analysis of extraordinarily large data. *Statistica Sinica*, 24:1655–1684, 2014.

[9] P. Drineas, M. Magdon-Ismail, M.W. Mahoney, and D.P. Woodruff. Faster approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13:3475–3506, 2012.

[10] P. Drineas, M.W. Mahoney, S. Muthukrishnan, and T. Sarlos. Faster least squares approximation. *Numerische Mathematik*, 117:219–249, 2011.

[11] Petros Drineas, Michael W Mahoney, and S Muthukrishnan. Sampling algorithms for l_2 regression and applications. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1127–1136. Society for Industrial and Applied Mathematics, 2006.

[12] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[13] Jianqing Fan, Fang Han, and Han Liu. Challenges of big data analysis. *National science review*, 1(2):293–314, 2014.

[14] William Fithian and Trevor Hastie. Local case-control sampling: Efficient subsampling in imbalanced data sets. *Annals of statistics*, 42(5):1693, 2014.

[15] Lei Han, Kean Ming Tan, Ting Yang, and Tong Zhang. Local Uncertainty Sampling for Large-Scale Multi-Class Logistic Regression. *The Annals of Statistics*, in press.

[16] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pages 315–323, 2013.

[17] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

[18] Matthias Katzfuss. A multi-resolution approximation for massive spatial datasets. *Journal of the American Statistical Association*, 112(517):201–214, 2017.

[19] Jerome H Klotz. Updating simple linear regression. *Statistica Sinica*, pages 399–403, 1995.

[20] Doug Laney. 3d data management: Controlling data volume, velocity and variety. *META group research note*, 6(70):1, 2001.

[21] Faming Liang, Yichen Cheng, Qifan Song, Jincheol Park, and Ping Yang. A resampling-based stochastic approximation method for analysis of large geostatistical data. *Journal of the American Statistical Association*, 108(501):325–339, 2013.

[22] Nan Lin and Ruibin Xie. Aggregated estimating equation estimation. *Statistics and Its Interface*, 4:73–83, 2011.

[23] Lan Luo and Peter X-K Song. Renewable estimation and incremental inference in generalized linear models with streaming data sets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82(1):69–97, 2020.

[24] Ping Ma, Michael W Mahoney, and Bin Yu. A statistical perspective on algorithmic leveraging. *The Journal of Machine Learning Research*, 16(1):861–911, 2015.

[25] Ping Ma, Xinlian Zhang, Xin Xing, Jingyi Ma, and Michael W Mahoney. Asymptotic analysis of sampling estimators for randomized numerical linear algebra algorithms. *arXiv preprint arXiv:2002.10526*, 2020.

[26] Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

[27] Per-Gunnar Martinsson and Joel Tropp. Randomized numerical linear algebra: Foundations & algorithms. *arXiv preprint arXiv:2002.01387*, 2020.

[28] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in neural information processing systems*, pages 1017–1025, 2014.

[29] Elizabeth D. Schifano, Jing Wu, Chun Wang, Jun Yan, and Ming-Hui Chen. Online updating of statistical inference in the big data setting. *Technometrics*, 58(3):393–403, 2016.

[30] Zuofeng Shang and Guang Cheng. Computational limits of a distributed algorithm for smoothing spline. *The Journal of Machine Learning Research*, 18(1):3809–3845, 2017.

[31] Panos Toulis, Edoardo M Airoldi, et al. Asymptotic and finite-sample properties of estimators based on stochastic gradients. *The Annals of Statistics*, 45(4):1694–1727, 2017.

[32] Chong Wang, Xi Chen, Alexander J Smola, and Eric P Xing. Variance reduction for stochastic gradient optimization. In *Advances in Neural Information Processing Systems*, pages 181–189, 2013.

[33] Chun Wang, Ming-Hui Chen, Jing Wu, Jun Yan, Yuping Zhang, and Elizabeth Schifano. Online updating method with new variables for big data streams. *Canadian Journal of Statistics*, 46(1):123–146, 2018.

[34] HaiYing Wang. Divide-and-conquer information-based optimalsubdata selection algorithm. *Journal of Statistical Theory and Practice*, 13(3):46, Jul 2019.

[35] HaiYing Wang. More efficient estimation for logistic regression with optimal subsamples. *Journal of Machine Learning Research*, 20(132):1–59, 2019.

[36] HaiYing Wang and Yanyuan Ma. Optimal subsampling for quantile regression in big data. *Biometrika*, Accepted.

[37] HaiYing Wang, Min Yang, and John Stufken. Information-based optimal subdata selection for big data linear regression. *Journal of the American Statistical Association*, 114(525):393–405, 2019.

[38] HaiYing Wang, Rong Zhu, and Ping Ma. Optimal subsampling for large sample logistic regression. *Journal of the American Statistical Association*, 113(522):829–844, 2018.

[39] Yan Wang, Chuan Hong, Nathan Palmer, Qian Di, Joel Schwartz, Isaac Kohane, and Tianxi Cai. A fast divide-and-conquer sparse Cox regression. *Biostatistics*, 09 2019. kxz036.

[40] Yining Wang, Adams Wei Yu, and Aarti Singh. On computationally tractable selection of experiments in measurement-constrained regression models. *The Journal of Machine Learning Research*, 18(1):5238–5278, 2017.

[41] Rui Xie, Zengyan Wang, Shuyang Bai, Ping Ma, and Wenxuan Zhong. Online decentralized leverage score sampling for streaming multidimensional time series. *Proceedings of machine learning research*, 89:2301, 2019.

[42] Yaqiong Yao and HaiYing Wang. Optimal subsampling for softmax regression. *Statistical Papers*, pages 585–599, 12 2018.

[43] Jun Yu, HaiYing Wang, Mingyao Ai, and Huiming Zhang. Optimal distributed subsampling for maximum quasi-likelihood estimators with massive data. *Journal of the American Statistical Association*, Accepted.

[44] Yuchen Zhang, John Duchi, and Martin Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *The Journal of Machine Learning Research*, 16(1):3299–3340, 2015.

[45] Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pages 1–9, 2015.