Chapter 14
Effective Statistical Methods for Big Data Analytics

Cheng Meng
University of Georgia, USA

Ye Wang
University of Georgia, USA

Xinlian Zhang
University of Georgia, USA

Abhyuday Mandal
University of Georgia, USA

Wenxuan Zhong
University of Georgia, USA

Ping Ma
University of Georgia, USA

ABSTRACT

With advances in technologies in the past decade, the amount of data generated and recorded has grown enormously in virtually all fields of industry and science. This extraordinary amount of data provides unprecedented opportunities for data-driven decision-making and knowledge discovery. However, the task of analyzing such large-scale dataset poses significant challenges and calls for innovative statistical methods specifically designed for faster speed and higher efficiency. In this chapter, we review currently available methods for big data, with a focus on the subsampling methods using statistical leveraging and divide and conquer methods.

1. INTRODUCTION

The rapid development of technologies in the past decade has enabled researchers to generate and collect data with unprecedented sizes and complexities in all fields of science and engineering, from academia to industry. These data pose significant challenges on knowledge discovery. We illustrate these challenges with examples from three different areas below.

- **Higgs Boson Data:** Discovery of the long-awaited *Higgs boson* was announced in July 2012 and was confirmed six months later, leading to a Nobel Prize awarded in 2013 (www.nobelprize.org). A Toroidal LHC Apparatus (ATLAS), a particle detector experiment constructed at the Large Hadron Collider (LHC) in The European Organization for Nuclear Research (CERN) is one of the

DOI: 10.4018/978-1-5225-2498-4.ch014

two LHCs that confirmed the existence of Higgs boson. The ATLAS generates the astronomically large amount of raw data about particle collision events, roughly one petabyte of raw data per second (Scannicchio, 2010). To put it into more tangible terms, one petabyte is enough to store the DNA of the entire population of the USA; one petabyte of average MP3-encoded songs (on mobile phones, roughly one megabyte per minute) would require 2,000 years to play. However, the analysis of the data at the scale of even tens or hundreds of petabytes is almost unmanageable using conventional techniques since the computation cost becomes intimidating or even not affordable at all.

- **Biological Experiments:** RNA-Seq experiments have been used extensively to study transcriptomes (Mortazavi et al., 2008, Nagalakshmi et al., 2008). They serve as one of the best tools so far for novel transcripts detection and transcript quantification in ultra-high resolution, by obtaining tens of millions of short reads. When mapped to the genome and/or to the contigs, RNA-Seq data are summarized by a super-large number of short-read counts. These counts provide a digital measure of the presence and/or prevalence of transcripts under consideration. In any genome-wide analysis, such as the bias correction model proposed by (Li et al., 2010), the sample size goes easily to millions, which renders the standard statistical computation infeasible.
- **State Farm Distracted Driver Detection Data:** Huge datasets are often generated by commercial companies nowadays. A dataset has been released by *State Farm*, the insurance company. *State Farm* is interested in testing whether dashboard cameras can automatically detect drivers engaging in distracted behaviors. Two-dimensional dashboard driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, applying makeups, reaching behind, *etc.*) are provided. The goal of statistical analysis is to predict the likelihood of what the driver is doing in the picture, i.e. whether computer vision can spot each driver's distracted behavior, such as *if they are not driving attentively, not wearing their seatbelt, or taking a selfie with their friends in the backseat*. In this case, the complexity of big data, i.e. the raw data being in the form of images, poses the first problem before performing any statistical analysis: converting imaging data into the matrix form is needed. In this example, the testing data itself consists of 22,424 images of 26 drivers in 10 scenarios, each with 60 to 90 images, and totaling the size of about 5 GB. The explosion of data generated can be imagined as the time recorded and the number of drivers increases.

This implication of big data goes well beyond the above. Facebook and Twitter generate millions of posts every second; Walmart stores and Amazon are recording thousands of millions of transactions 24 hours 7 day, *etc*. Super large and complicated data sets provide us with unprecedented opportunities for data-driven decision-making and knowledge discoveries. However, the task of analyzing such data calls for innovative statistical methods for addressing the new challenges emerging every day due to the explosion of data.

Without loss of generality, in the rest of this chapter, we will assume that the datasets are already converted to numerical forms. Different statistical techniques will be discussed for analyzing large datasets. These datasets are so large that standard statistical analysis cannot be performed on a typical personal computer (PC). From a statistical point of view, the large data could arise in the following cases, either huge numbers of predictors, huge numbers of sample size or both. In what follows, we will focus on the second scenario. Next, we present the engineering solutions to this problem, point out the advantages and disadvantages, and then introduce the statistical solutions.

1.1. Engineering Solutions

For computer engineers, a straightforward way to reduce computing time is to resort to more powerful computing facilities. Great efforts have been made to solve the problem of big data by designing supercomputers. Many supercomputers have been built rapidly in the past decade, such as Tianhe-2, Bluewater and Blue Gene (Top500.org, 2014). The speed and storage of supercomputers can be hundreds or even thousands of times faster and larger compared to that of a general-purpose PC. However, the main problem with supercomputers is that they consume enormous energy and are not accessible to ordinary users. Thus, although supercomputers can easily deal with large amounts of data very efficiently, they are still not a panacea. Instead, cloud computing can partially address this problem and make computing facilities accessible to ordinary users. Nonetheless, the major bottleneck encountered by cloud computing is the inefficiency of transferring data due to the precious low-bandwidth internet uplinks, not to mention the problems of privacy and security concerns during the transfer process (Gai and Li, 2012). Another relatively new computational facility proposed is the graphic processing unit (GPU), which is powerful in parallel computing. However, a recently conducted comparison found that even high-end GPUs are sometimes outperformed by general-purpose multi-core processors, mainly due to the huge data transferring time (Pratas et al., 2012). In brief, none of the supercomputer, the cloud computing, GPUs solves the big data problem efficiently at this point (Chen and Zhang, 2014). Efficient statistical solutions are required, which makes big data problem manageable on general-purpose PCs.

1.2. Statistical Solutions

The statistical solutions are relatively novel compared to the engineering solutions. New methodologies are still under development. The methods available now can broadly be categorized into three groups: (1) divide and conquer method; (2) fine to coarse method; (3) sampling method. To be specific, we set our context as a dataset of n identically distributed observations and one response variable with p explanatory variables. Our statistical goal will be set for Model Estimation now.

1.2.1. *Divide and Conquer Method*

The divide and conquer method solves big data problems in the following manner. First, the original big dataset is divided into K small blocks that are manageable to the current computing facility unit. Then, the intended statistical analysis is performed on each small block. Finally, an appropriate strategy will be used to combine the results from these K blocks. As a result, the computation for the divide and conquer method can easily be done in parallel. However, the challenge lies in providing strategies for combining the results from smaller blocks. This is trivial for some models, like linear models or generalized linear models, for which the estimation procedures are linear by construction. More specifically, the estimating equations for the full data themselves can be written as a summation of all smaller blocks. The readers are referred to (Li et al., 2013) for more detailed discussion and theoretical properties for resulting estimators for a single parameter case. For other models, including but not limited to nonlinear parametric models (Lin and Xi, 2011), nonparametric models based on kernel regression (Xu et al., 2015), and penalized generalized linear regression models (Chen and Xie, 2014), the divide and conquer method in general still lacks a universal combining strategy which can handle all these cases.

1.2.2. Fine to Coarse Method

Another surprising yet proved to be effective idea proposed much recently is the Fine to Coarse method. To make intended algorithms for the big dataset scalable, statisticians introduced a simple solution: rounding parameters. Hence the continuous real numbers of data are simply rounded from higher decimal places to lower decimal places. A substantial number of observations are degenerated to be identical. This idea was successfully applied to the functional data analysis using smoothing spline ANOVA models. See (Helwig and Ma, 2016) for more details.

1.2.3. Sampling Methods

Another more effective and more general solution to the big data problem is the sampling method. That means that we take a subsample from the original dataset with respect to a carefully designed probability distribution, and use this sample as a surrogate for the original dataset to do model estimation, predictions as well as statistical inference. The most important component of this method is the design of probability distribution for taking the sample.

One naïve choice for the probability distribution is the simple uniform distribution. If we further set the subsample size as *n*, then it reduces to the procedure of bootstrap (Efrom, 1979, Wu, 1986, Efron, 1992, Shao and Tu, 2012). On the other hand, a great deal of efforts has been spent on developing algorithms for matrix-based machine learning methods and data analyses that construct the random sample in a non-uniform data-dependent fashion (Mahoney, 2011). In particular, a large body of literature specifically pointed out that the subsampling probability distribution using the statistical leverage scores outperforms uniform sampling for different purposes, especially in matrix approximation related problems (Drineas et al., 2006, Mahoney and Drineas, 2009, Drineas et al., 2011). Furthermore, efforts were put on studying the performance of leveraging based estimators from a statistical point of view (Ma et al., 2015, Ma and Sun, 2015).

Overall, the main advantage of the sampling method is its general application to various model settings. Moreover, it will automatically give rise to a random sketch of the full data as a byproduct, which is useful for the purpose of data visualization. However, the nontrivial part of using sampling method is the construction of sampling probability distribution, which plays a crucial role in sampling methods. The rest of this chapter is dedicated to elaborate on the different designs of sampling probability distributions.

2. STATISTICAL FORMULATION OF BID DATA PROBLEM

In this section, we first introduce some general background of the linear model, then discuss the general sampling method which deals with the linear model problem in big data.

2.1. Classical Linear Regression Model

Throughout the chapter, we define y as the response vector, X as the predictor matrix, n as the number of data points and p as the dimension of the predictors.

We start with the classical linear regression model:

$$y = X\beta + \varepsilon \tag{1}$$

where y is a n×1 vector, X is a n×p matrix consisting of one intercept and p–1 explanatory variables and β is the p×1 coefficient vector, ε is the noise term which is assumed to follow a multivariate normal distribution N(0, δ2^{I}).

In linear models, the coefficient vector β can be estimated by calculating the ordinary least square (OLS), that is:

$$\widehat{\beta}_{OLS} = \arg\min_{\beta} \left\| y - X\beta \right\|^2 \tag{2}$$

where $\left\|.\right\|$ represents the Euclidean norm on the n-dimensional Euclidean space R^n. When X is full column rank, it can be shown that:

$$\widehat{\beta}_{OLS} = \arg\min_{\beta} \left\| y - X\beta \right\|^2 = \left(X^T X\right)^{-1} X^T y \tag{3}$$

Otherwise, when X is singular, $(X^TX)^{-1}$ should be replaced by a generalized inverse of X^TX. Consequently, the predicted response vector $\hat{y}$ can be represented as:

$$\hat{y} = X\left(X^T X\right)^{-1} X^T y \tag{4}$$

The projection matrix, $X(X^TX)^{-1}X^T$, is often referred to the hat matrix H since it looks like a hat on response vector y to get $\hat{y}$. The hat matrix H plays a crucial role in the subsequent analysis in Section 3.

To get predicted response $\hat{y}$, it suffices to calculate H, i.e. $X(X^TX)^{-1}X^T$. For robustness concern, people usually carry out the required computations by using the singular value decomposition (SVD) instead of calculating the matrix inverse directly (Golub & Van Loan, 2012).

Through some calculations, it can be shown that $H=UU^T$, $\hat{\beta}_{OLS} = \left(X^T X\right)^{-1} X^T y = V\Lambda^{-1}U^T y$.

Box 1. Singular Value Decomposition

Given any $n\times p$ matrix X, we can always decompose it to the form

$$X_{n\times p} = U_{n\times n}\Lambda_{n\times p}V^T_{p\times p},$$

where U and V are both orthonormal matrices and Λ is a diagonal matrix with all the singular values of X on the diagonal.

Algorithm 1: *General Sampling Method in Linear Model*

Step 1 (Subsampling): Take a random sample of size *r*>*p* from the full data based on a sampling probability distribution $\left\{\pi_i\right\}_{i=1}^{n}$ such that $\sum_{i=1}^{n}\pi_i = 1, 0 < \pi_i < 1$. Record the chosen data as $\left\{y_i^*, X_i^*\right\}_{i=1}^{r}$, along with the sampling probabilities for the chosen data $\left\{\pi_i^*\right\}_{i=1}^{r}$.

Step 2 (Model-fitting): Use the subsample to fit a weighted least square with weight $\left\{1/\pi_i^*\right\}_{i=1}^{r}$ and obtain the estimator $\tilde{\beta}$ as follows:

$$\tilde{\beta} = argmin_{\beta}\left(y^* - X^*\beta\right)^T W\left(y^* - X^*\beta\right),$$

where $W = Diag\left(\left\{1/\pi_i^*\right\}_{i=1}^{r}\right)$

2.2. General Sampling Method

As mentioned before, in the sampling approach we first choose a small subset of the full data, which we term as "subsampling step," then use this sample to estimate the model parameters, which we term as "model-fitting step." In the linear model setup, this approach can be utilized by sampling a small portion of rows from the input matrix *X* and then by carrying out linear regression on the sample data. Putting this idea in the framework of the linear model, we come up with the Algorithm 1.

Remark 1: One may wonder why the weighted least square (WLS) instead of ordinary least square (OLS) is used in the second step. This is because the estimator resulting from Algorithm 1 is a conditional asymptotically unbiased estimator for $\hat{\beta}_{OLS}$, i.e., $E\left(\tilde{\beta}|data\right) \approx \hat{\beta}_{OLS}$ and it is also an unbiased estimator for the true parameter, i.e. $E\left(\tilde{\beta}\right) = \beta$ (Ma et al., 2014, Ma et al., 2015). If OLS instead of WLS is used in the second step, the conditional asymptotically unbiasedness property will be lost. However, in that process of pertaining the unbiasedness, one can potentially end up with an estimator with a higher variance. More insights into the gains and losses for estimators result from weighted and unweighted least square estimation for subsample data will be given in Section 4.

Remark 2: Although not explicitly described, asymptotic evaluation of $\tilde{\beta}$ shows that the sampling probability distribution $\left\{\pi_i\right\}_{i=1}^{n}$ plays an essential role in the property of the resulting $\tilde{\beta}$, especially in estimating the variance of $\tilde{\beta}$. The main goal of the rest of the chapter is to propose a computationally efficient design of $\left\{\pi_i\right\}_{i=1}^{n}$ for better estimation accuracy.

Algorithm 2: *Uniform Sampling Method in Linear Model*

Step 1 (Subsampling): Take a random sample of size $r>p$ from the full data using a uniform sampling distribution and denote the subsample as $\left\{y_i^*, X_i^*\right\}_{i=1}^r$.

Step 2 (Model-fitting): Using the subsample to fit the least square, obtain the estimator $\tilde{\beta}_{UNIF}$ as follows:

$$\tilde{\beta}_{UNIF} = argmin_{\beta} \left\| y^* - X^* \beta \right\|^2$$

3. LEVERAGE-BASED SAMPLING METHOD

In this section, we introduce two examples of the general sampling methods, the Uniform Sampling Method and Basic Leverage Sampling Method, and give illustrations on the advantages as well as disadvantages of both algorithms.

3.1. Uniform Sampling Method

The most naïve version of the sampling method is to apply the Algorithm 2 with uniform probabilities, i.e. $\pi_i = \frac{1}{n}$, for i=1,2,…,n. In this particular situation, the WLS in step 2 reduces to the OLS.

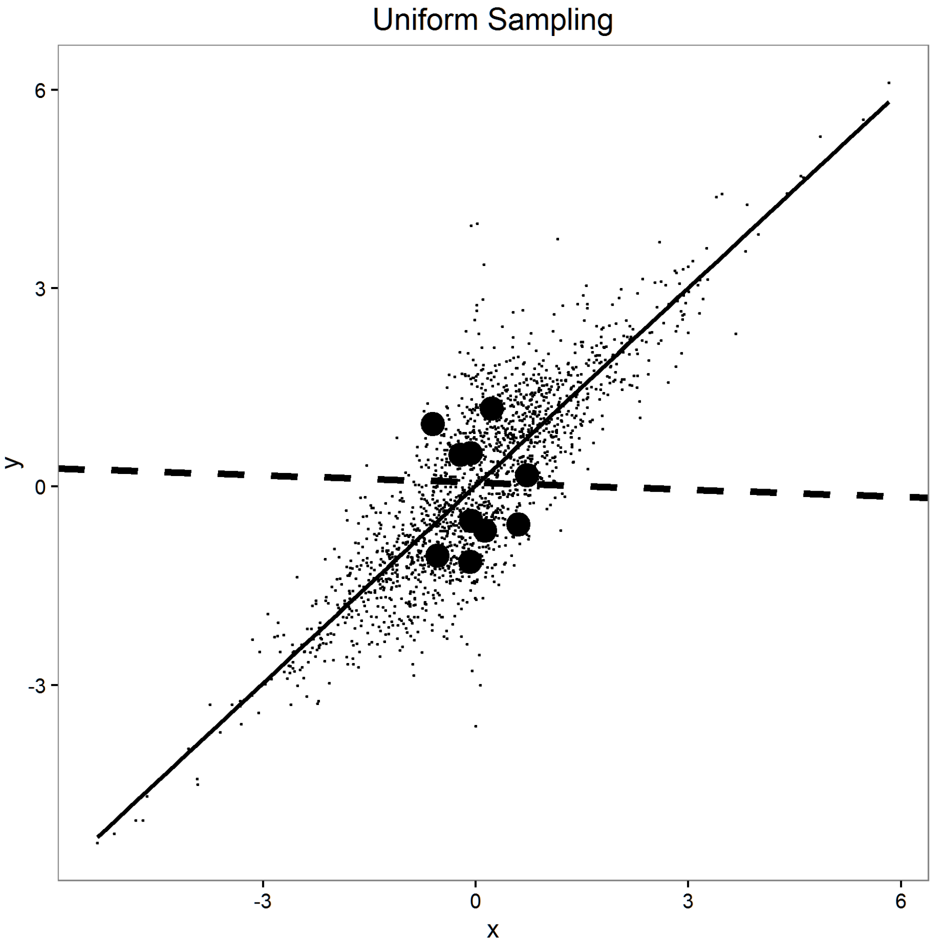
This algorithm is easy to understand. Instead of using full data to calculate the least square estimator, we just take a simple random sample from it and calculate the estimator $\tilde{\beta}_{UNIF}$ using the subsample. One obvious advantage of this algorithm is the short computing time, which is $O(rp^2)$. Another advantage, as we mentioned in Remark 1 of Algorithm 1, is the unbiasedness of $\tilde{\beta}_{UNIF}$. However, as implied in Remark 2 of Algorithm 1, the large variance of the estimator is the main drawback of this method. When the sampling size r is small, there is a good chance that the estimator $\hat{\beta}_{UNIF}$ can be totally different from $\hat{\beta}_{OLS}$. This situation can be illustrated by the following example.

In Figure 1, the data points sampled from uniform probabilities did not identify the main linear pattern of the full data, which caused a big difference between $\tilde{\beta}_{UNIF}$ and $\hat{\beta}_{OLS}$. This significant difference is because Uniform Sampling Method ignores the different contribution of different data points for estimating $\hat{\beta}_{OLS}$. A good sampling strategy should take these differences into account. For example, if the subsampled points are spread out, that is, points in the upper right and lower left corners are included, then the fitted line will be much closer to the "truth." Since those points in the upper and lower corner of Figure 1 are high leverage points, it is easy to understand the motivation of the leverage-based sampling method discussed below.

3.2. Leverage Score and Basic Leverage Sampling Method

In the previous subsection, we mentioned that we needed to find the data points that are influential for fitting the regression line. In the statistical literature for model diagnostics, there exists the concept of leverage score to achieve a similar goal (Weisberg, 2005). For the ith data point (y_i, X_i), we define the

Figure 1. Example of the failure of the Uniform Sampling Method. For i=1,...,2000, $y=-0.5+x_i+\varepsilon_i$, where x_i is generated from t-distribution with df =6 and ε_i~N(0,1). The small dots are the original data points; big dots are the subsample points. The solid line represents the fitted regression line of the full data, and the dashed line represents the fitted regression line of the subsamples.



leverage score as $\frac{\partial \hat{y}_i}{\partial y_i}$. Intuitively, if the leverage score is large, it means that a small disturbance in y_i results in a big change in $\hat{y}_i$, thus playing a crucial role in model diagnostics.

There is also an elegant explanation for this definition. In Section 2.1, we mentioned about the "hat matrix" H which follows the relationship $\hat{y} = Hy$, i.e.

$$\begin{pmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \cdots \\ \hat{y}_n \end{pmatrix} = \begin{pmatrix} h_{11} & \dots & h_{1n} \\ h_{21} & \dots & h_{2n} \\ \vdots & \ddots & \vdots \\ h_{n1} & \cdots & h_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} \tag{5}$$

Using this relationship, the leverage score can be written as:

$$\frac{\partial \hat{y}_i}{\partial y_i} = \frac{\partial \left(\sum_{j=1}^{n} h_{ij} y_j \right)}{\partial y_i} = h_{ii}. \tag{6}$$

Hence, the leverage score for the *i*th data point is just the *i*th diagonal element of hat matrix *H*.

Also, it is easy to show that $Var\left(e_i\right) = Var\left(\hat{y}_i - y_i\right) = \left(1 - h_{ii}\right)\sigma^2$, which means the high leverage points have small variances of residuals and that in general $0<h_{ii}<1$. This result shows that the regression line tends to pass close to these data points with high leverage scores, indicating their large impact on the regression line. For example, in the univariate linear model, where the design matrix *X* can be written as

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \dots & \dots \\ 1 & x_n \end{bmatrix} \tag{7}$$

$$h_{ii} = \frac{1}{n} + \frac{\left(x_i - \bar{x}\right)^2}{\sum_{j=1}^{n} \left(x_j - \bar{x}\right)^2},$$

where $\bar{x} = \frac{\sum_{j=1}^{n} x_j}{n}$. In this particular case, the data points with large leverage scores are the data points far away from the mean of the full data, like the points in the upper right corner and lower left corner of Figure 1, confirming our previous guess. This result also meets the general understanding of a high influential point.

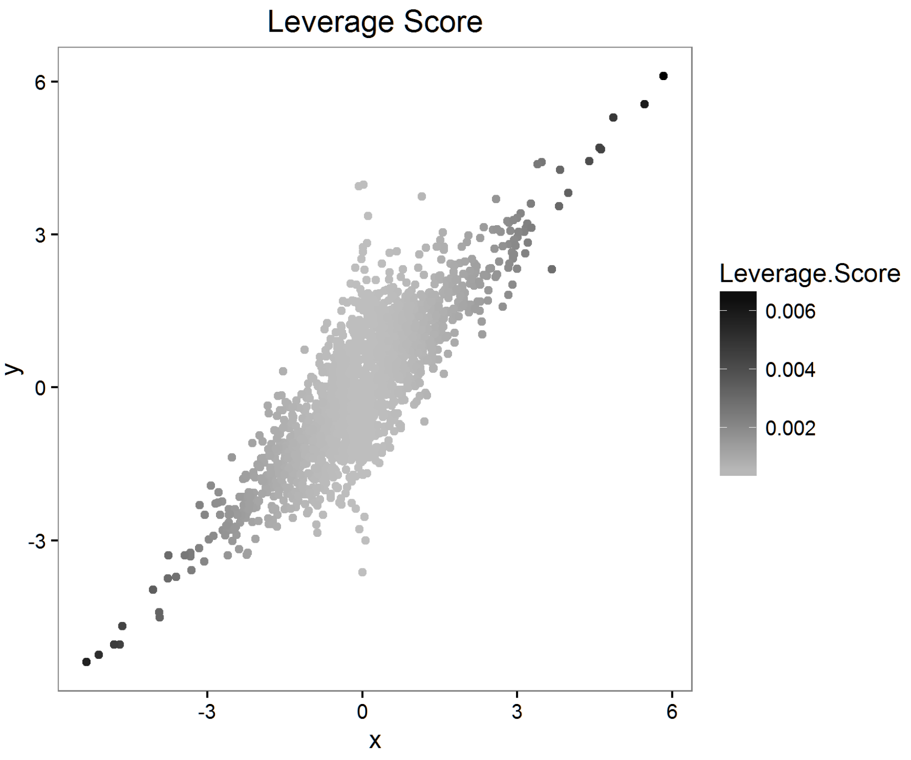
When the model matrix *X* is full column rank, the sum of all the leverage scores of the full data is just the dimension *p*. Hence, $0 < \pi_i^{BLEV} = \frac{h_{ii}}{p} < 1$ with $\sum_{i=1}^{n} \pi_i^{BLEV} = 1$, since

$$\sum_{i=1}^{n} h_{ii} = tr\left(H\right) = tr\left(X\left(X^T X\right)^{-1} X^T\right) = tr\left(\left(X^T X\right)^{-1} X^T X\right) = tr\left(I_p\right) = p \tag{8}$$

These facts motivate the Basic Leverage Sampling Method (BLEV) discussed in Algorithm 3.

This Basic Leverage Sampling Method is another application of the General Sampling Method, in which the sampling probability are substituted by the probability distribution constructed from leverage scores. The computational complexity for BLEV is $O(np^2)$. Same as $\tilde{\beta}_{UNIF}$, $\tilde{\beta}_{BLEV}$ is also a conditional asymptotically unbiased estimator of $\tilde{\beta}_{OLS}$ (Ma et al., 2014, Ma et al., 2015). An example of the BLEV is shown in Figure 3.

Figure 2. Illustration of the leverage scores of the data points from the example in Figure 1. In a univariate linear model, the data points far away from the mean have higher leverage scores.



Algorithm 3: *Basic Leverage Sampling Method in Linear Model*

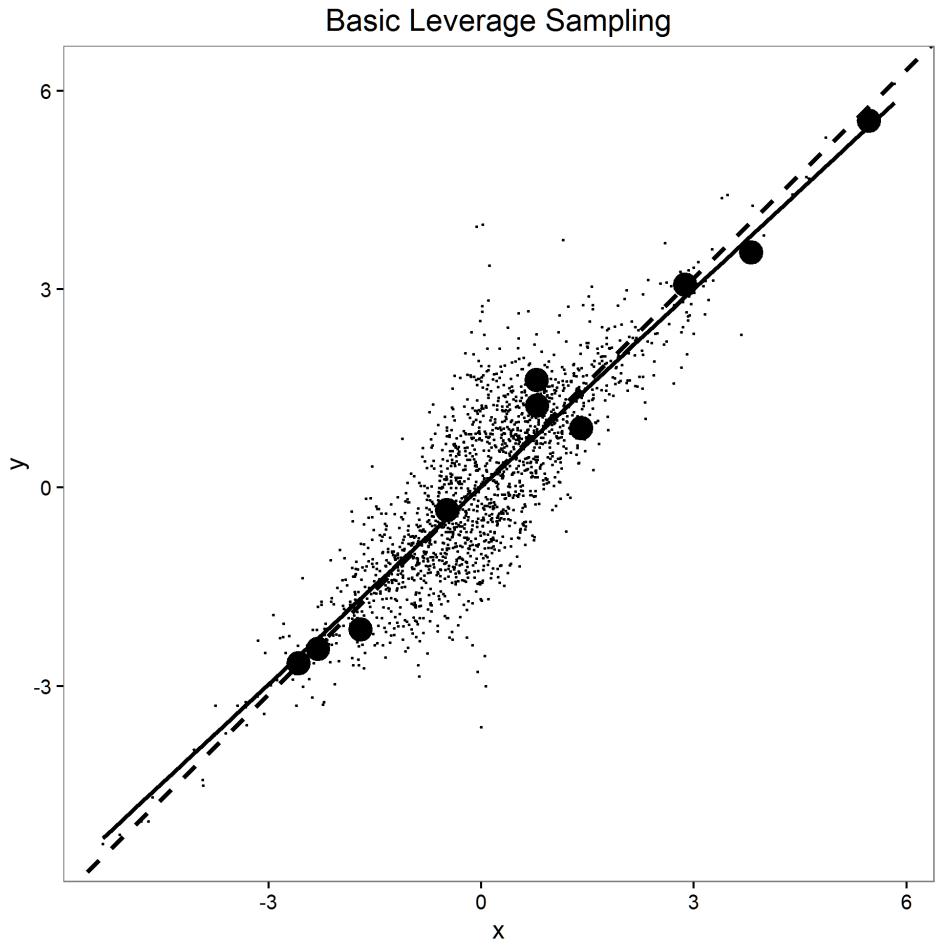
Step 1 (Subsampling): Take a random sample of size *r>p* from the full data using the probability distribution $\left\{\pi_i^{BLEV}\right\}_{i=1}^{n} = \left\{\frac{h_{ii}}{p}\right\}_{i=1}^{n}$ and denote it as $\left\{y_i^*, X_i^*\right\}_{i=1}^{r}$. Record the corresponding sampling probability as $\left\{\pi_i^*\right\}_{i=1}^{r}$.

Step 2 (Model-fitting): Use the subsample to fit a weighted least square with weight $\left\{1 / \pi_i^*\right\}_{i=1}^{r}$ and obtain the estimator $\tilde{\beta}_{BLEV}$.

Compared to Figure 1, the advantage of BLEV is obvious, since the fitted regression line of the leverage sub-samples is very close to the fitted regression line of the full data. The probability that this scenario occurs equals to the multiplication of the leverage sampling probabilities for the sub-samples, which is 2×10^{-30} in this case. This sub-sample is relatively unlikely to be sampled from uniform probability distribution, since $(1/2000)^{10}=1\times10^{-33}$. In contrast, the sub-sample in Figure 1 is relatively unlikely to be sampled from leverage probability distribution, since the multiplication of the leverage sampling probabilities for the sub-samples in Figure 1 equals to 3.7×10^{-37}, which is much smaller than $(1/2000)^{10}$.

From the example in Figure 1 and Figure 3, the Basic Leverage Sampling Method can be utilized to solve linear model problems in big data intuitively.

Figure 3. An example of the Basic Leverage Sampling Method. The data are the same that in Figure 1. The small dots are the original data; the big dots are the sample. The solid line represents the fitted regression line of the full data, and the dashed line represents the fitted regression line of the subsamples.



3.3. Disadvantage of Basic Leverage Sampling Method

From the observation in the last subsection, one may assume that BLEV should always have better performance than UNIF. This seems to be true in algorithmic point of view. Prior work has adopted an algorithmic perspective that focuses on providing worst-case run-time bounds for different inputs. It has been shown that leverage-based sampling provides worst-case algorithm results that are uniformly superior to the uniform sampling method (Drineas et al., 2006). However, in a statistical point of view, neither BLEV nor UNIF dominates the other (Ma et al., 2014, Ma et al., 2015). Actually, it has been shown that the variance of estimator $\tilde{\beta}_{BLEV}$ may be inflated by extremely small leverage scores. This could happen when the data distribution has a heavy tail, e.g. a Student-t distribution with small degree of freedom or Cauchy distribution. In such cases, the data points on the tail tend to have enormous leverage scores which dominate the others. For example, consider the case when the dataset has a different distribution in each dimension, the Basic Leverage Sampling Method may fail to capture all the high-influential points. Such a case can be illustrated by the example in Figure 4.

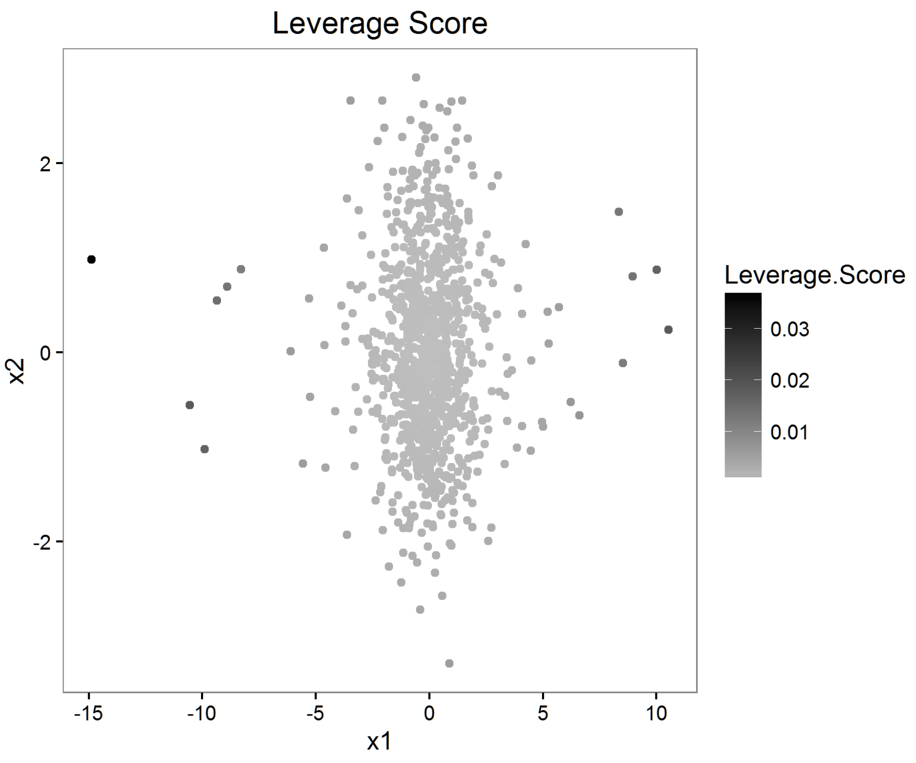
As shown in Figure 4, the high-leverage-score points are only the high influential points for the first dimension, but not for the second dimension. Hence, the subsamples chosen by BLEV are inadequate to predict the second dimension of $\tilde{\beta}_{OLS}$, leading to a bad estimator $\tilde{\beta}_{BLEV}$. That indicates that simply using the leverage score as a sampling probability seems too aggressive.

Furthermore, it poses the interesting question of if we could try different subsampling probabilities and propose even better sampling method than BLEV. These will be discussed in the next section.

4. NOVEL LEVERAGING-BASED SAMPLING METHOD

In this section, we will introduce two novel Leverage-Based Sampling Methods which aims at overcoming the drawback of BLEV.

Figure 4. Illustration of the leverage scores of the 1000 data points from a two-dimensional data set, which the first dimension comes from a Student-t distribution with degree of freedom 4 and the second dimension comes from a standard normal distribution.



4.1. Shrinkage Leveraging Method

Recall that we want to give large sampling probabilities to the points with large leverage scores since these points are more influential to the fitted regression line. Using the leverage score as the sampling probability is a simple way to accomplish this goal, which generates the Basic Leverage Sampling Method.

In fact, as long as we preserve the ranking of the leverage score, we can still take the benefit of the influential points. We can achieve this goal by comparison between applying the following relatively conservative probability distribution SLEV (shrinkage leveraging) and applying the basic leverage score distribution. Let π_i^{BLEV} denote a distribution defined by the basic leverage scores (i.e., $\pi_i^{BLEV} = \frac{h_{ii}}{p}$) and let $\pi_i^{UNIF} = \frac{1}{n}$ denote the uniform distribution; then the sampling probabilities for the shrinkage leveraging can be written as:

$$\pi_i^{SLEV} = \alpha\pi_i^{BLEV} + \left(1-\alpha\right)\pi_i^{UNIF}, \alpha \in \left(0,1\right) \tag{9}$$

for $i=1,2,\ldots,n$.

Applying these sampling probabilities leads us to the Shrinkage Leverage Sampling Method (SLEV).

The computational complexity for SLEV is $O(np^2)$. The performance of the Shrinkage Leverage Sampling Method depends on how we choose the shrinkage index α. If we choose α very close to 0 or 1, it will just degenerate into the uniform sampling method or basic leverage sampling method. However, if we choose the α more wisely, the SLEV method can overcome the large variance problem. The recommended value of α is falling in the interval [0.8, 0.9] (Ma et al., 2014, Ma et al., 2015). Under this situation, the SLEV can preserve the ranking of the leverage score without containing extremely large or extremely small sampling probabilities, compared to Basic Leverage Sampling probabilities. Furthermore, all these observations also hold if we use the approximate leverage score instead of using the exact leverage score in the method. For these reasons, the SLEV procedure with approximate leverage score is the most recommended method for linear models in big data.

4.2. Unweighted Leverage Sampling Method

Before introducing the Unweighted Leverage Sampling Method, we need to discuss the criteria for judging whether a particular sampling method is good or not. From a statistical point of view, we need a

Algorithm 4: *Shrinkage Leverage Sampling Method in Linear Model*

Step 1 (Subsampling): Take a random sample of size $r>p$ from the full data using the probability distribution $\left\{\pi_i^{SLEV}\right\}_{i=1}^{n}$ and denote it as $\left\{y_i^*, X_i^*\right\}_{i=1}^{r}$. Record the corresponding sampling probability as $\left\{\pi_i^*\right\}_{i=1}^{r}$.

Step 2 (Model-fitting): Use the subsample to fit a weighted least square with weight $\left\{1/\pi_i^*\right\}_{i=1}^{r}$ and obtain the estimator $\tilde{\beta}_{SLEV}$.

comprehensive criterion to consider both bias and variance simultaneously. The mean squared error (MSE) is a reasonable choice. The formula for MSE for $\tilde{\beta}$ is given below.

$$MSE\left(\tilde{\beta} \mid y\right) = E\left\|\tilde{\beta} - \hat{\beta}_{OLS}\right\|^2 \tag{10}$$

Some decomposition analysis will give that

$$MSE\left(\tilde{\beta}\right) = \left\|Bias\left(\tilde{\beta}\right)\right\|^2 + tr\left(Var\left(\tilde{\beta}\right)\right) \tag{11}$$

where we denote $Bias\left(\tilde{\beta}\right) = E\left(\tilde{\beta}\right) - \hat{\beta}_{OLS}$. This decomposition is sometimes termed bias-variance decomposition in the statistics literature.

We know that the estimator generated by UNIF, BLEV and SLEV are all unbiased estimators. This is a very appealing property, and we only need to focus on minimizing the variance of the estimator. However, if our goal is to minimize MSE, it is not necessary to let the estimator be asymptotically unbiased. In other words, an estimator with bias can still be a good estimator if it has a relatively small bias but significantly smaller variance. This is also the main motivation of Unweighted Leverage Sampling Method discussed in Algorithm 5.

It could be theoretically shown that the unweighted leverage estimator is an unbiased estimator to β as well as a conditionally unbiased estimator to the weighted least square estimator $\hat{\beta}_{WLS}$ conditional on given data (Ma et al., 2014, Ma et al., 2015). As a conditionally biased estimator, $\hat{\beta}_{WLS}$ is rarely a concern from an algorithmic perspective. However, from a statistician's point of view, the disadvantage brought by biasedness can be mitigated by the advantage by a significant decrease in variance if our main goal is to minimize MSE. This is exactly the main advantage of an unweighted leverage estimator compared to the Basic Leverage Sampling Method, i.e. it overcomes the inflated variance problem.

5. SOFTWARE IMPLEMENTATION

The key step of our BLEV, SLEV, LEVUNW method is the calculation of leverage scores a design matrix, i.e. applying SVD on it. Almost all the popular statistical software packages are available for this task such as command svd in R base, command svd in MATLAB, subroutine SVD from SAS. The

Algorithm 5: *Unweighted Leverage Sampling Method in Linear Model*

```
Step 1 (Subsampling): Take a random sample of size $r>p$ from the full data, probability distribution $\left\{\pi_i^{LEVUNW}\right\}_{i=1}^n = \left\{\frac{h_{ii}}{p}\right\}_{i=1}^n$ and denote it as $\left\{y_i^*, X_i^*\right\}_{i=1}^r$.
Step 2 (Model-fitting): Use the subsample to fit an ordinary least square and obtain the estimator $\tilde{\beta}_{LEVUNW}$.
```

Box 2. R Code

```
####################################################################
# First, we construct a univariate linear model and set the true
# beta vector as (10,5).
####################################################################
setseed=100
set.seed(setseed)
n = 10000
xx = rnorm(n)
y = 10+5*xx+rnorm(n)
####################################################################
# Second, we construct the predictor matrix X.
####################################################################
X = cbind(1,xx)
####################################################################
# Third, we perform SVD for matrix X. Then, we extract the U
# matrix of X. Using U, we extract the leverage scores of all
#observations and put in vector hii.
####################################################################
svdx = svd(X)
U = svdx$u
hii = apply(U,1,crossprod)
####################################################################
# We construct subsampling probability distribution for BLEV and
# SLEV.
####################################################################
blev.prob = hii/2
slev.prob = hii/2*0.9+1/n*0.1
####################################################################
# We set the subsample size r.
####################################################################
r = 100
####################################################################
# Next, perform subsampling using hii as subsampling probability
# distribution and record the subsampling probabilities of the
# subsampled data
####################################################################
blev.ind = sample.int(n=n,size=r,replace=TRUE,prob=blev.prob)
slev.ind = sample.int(n=n,size=r,replace=TRUE,prob=slev.prob)
y.blev = y[blev.ind]
y.slev = y[slev.ind]
xx.blev = X[blev.ind,]
xx.slev = X[slev.ind,]
wgt.blev = 1/blev.prob[blev.ind]
wgt.slev = 1/slev.prob[slev.ind]
####################################################################
# Now perform WLS on the subsampled data for BLEV and SLEV,
# perform OLS on the subsampled data for LEVUNW
####################################################################
lm.blev = lm(y.blev~xx.blev-1, weights = wgt.blev)
lm.slev = lm(y.slev~xx.slev-1, weights = wgt.slev)
lm.levunw = lm(y.blev~xx.blev-1)
bt.blev = lm.blev$coefficients
bt.slev = lm.slev$coefficients
bt.levunw = lm.levunw$coefficients
####################################################################
# In order to evaluate the performance of these sampling methods,
# we run the OLS for full data
####################################################################
lm.full = lm(y~X-1)
summary(lm.full)
bt = lm.full$coefficients
####################################################################
# Finally, we calculate the SE of estimator from this subsampled
# data.
####################################################################
SE_blev = crossprod(bt-bt.blev)
SE_slev = crossprod(bt-bt.slev)
SE_levunw = crossprod(bt-bt.levunw)
```

underlying source code for these procedures is all from LAPACK routines or equivalent. For illustration, we provide an R code.

Remark 3: When n gets large enough, calculating SVD poses a challenge in computer memory. In practice, QR decomposition is recommended instead of SVD in this case.

The order of computational cost of all the subsampling methods introduced in this chapter so far are dominated by the SVD of original data matrix X, which will be $O(np^2)$ using one of the earliest algorithms (Golub and Van Loan, 2012) and this is the same magnitude of the time order for solving the original linear problem with full data. Fortunately, there already exists fast approximation algorithms for leverage scores that can be used to achieve this goal, which decreases the running time from $O(np^2)$ to $o(np^2)$ (Drineas et al., 2012). In specific, given an arbitrary $n \times p$ matrix X such that $n \gg p$, and an error parameter $\varepsilon \in (0,1)$, the main algorithm of (Drineas et al., 2012) is based on random projection, and it computes $\tilde{l}_i$ as an approximation of the h_{ii} in the sense that $\left|\tilde{l}_i - h_{ii}\right| \leq \epsilon h_{ii}$ for all $i=1,\ldots,n$. This algorithm runs in roughly $O(np\log(p)/\varepsilon)$ time, which will be $o(np^2)$ under appropriate settings. See Blendenpik (Avron et al., 2010), LSRN (Meng et al., 2014) as well as (Gittens and Mahoney, 2013) for further upgrading of aforementioned random projection algorithms. It is documented in these studies that if the dimension of input matrix is at least as small as several thousand by several hundred, the run time of the leveraging-based methods can be competitive compared to solving the original linear problem by QR decomposition or SVD with e.g. LAPACK.

6. DEMONSTRATION: TWO CASE STUDIES

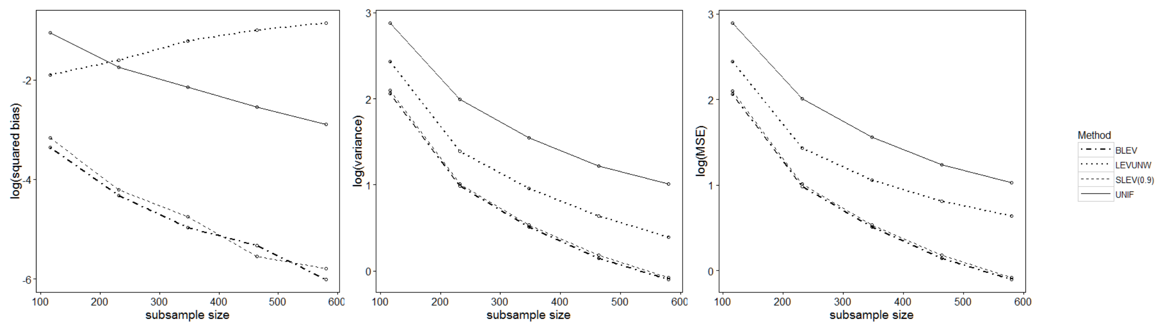
To illustrate the performance of the sampling methods on real data, two datasets are considered: an *Illumina HiSeq data set* downloaded from TCGA (http://cancergenome.nih.gov) and the "YearPredictionMSD" dataset, a subset of the Million Song Dataset (http://labrosa.ee.columbia.edu/millionsong/). The former has a strong linear pattern while the latter does not. This property of the dataset will influence the behavior of these methods.

Coefficient estimates were obtained using four subsampling algorithms (UNIF, BLEV, SLEV(0.9) and LEVUNW) for five different subsampling size: $2p$, $4p$, $6p$, $8p$, $10p$. The subsampling size is chosen based on the $n=10p$ rule, which proposed by (Loeppky et al., 2012). For each subsample size, we take 200 hundred subsamples and calculate estimates based on each of the subsampling algorithms. We then calculate the empirical conditional biases and variances with respect to the full sample least square estimate.

6.1. Illumina HiSeq Dataset

Considering an *Illumina HiSeq data set* downloaded from TCGA for 59 cancer patients which contain $n=20{,}529$ genes. Here, one patient's data are randomly chosen as the response y and use the remaining patients' data as the predictors through a linear model. Thus, the number of predictors in this setup is $p=58$. We first adopt a commonly used transformation for the counts data, i.e. $\log(X+1)$. After transforming the original data, we fit a linear model for the entire data. The adjusted-R^2 is 0.9879, which represents an almost perfect fit. Next, the dataset is fit to a linear model using subsampling methods with five different subsampling sizes. Figure 5 shows the summary of our results.

Figure 5. Empirical results for the Illumina HiSeq data set. The left panel is the empirical conditional squared biases of the UNIF, BLEV, SLEV, LEVUNW; middle panel is the empirical conditional variance; right panel is the empirical conditional MSE. Solid lines for UNIF; dash lines for BLEV; the thick dotted line for LEVUNW; the thin dotted line for SLEV with α= 0.9.



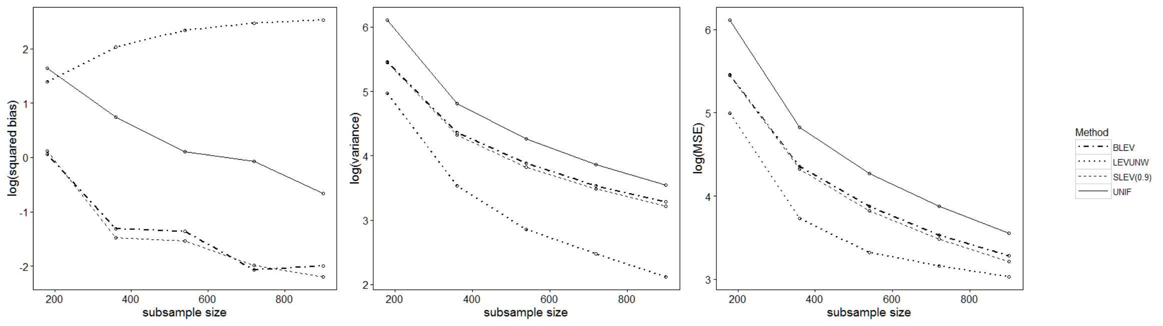
In the left panel of Figure 5, we plot the empirical conditional squared biases of the four methods. Observe that BLEV and SLEV both have smaller squared biases than UNIF and LUVUNW, which indicates that both BLEV and SLEV capture the main linear pattern of the whole dataset more efficiently than UNIF. As mentioned above, compared to $\hat{\beta}_{OLS}$, LUVUNW is a conditionally biased estimator. Thus, as the sample size becomes larger, the squared bias of LUVUNW does not decrease. Since the conditional variance, the dominant part of MSE, of LUVUNW is much smaller than that of UNIF, it still outperforms UNIF when MSE is our final consideration. In this example, BLEV and SLEV have almost the same performance and are consistently better than UNIF and LUVUNW. This is due to the strong linear pattern of the dataset. The phenomenon of a weak linear pattern of the dataset will be strongly influenced by the behavior of these sampling methods could be seen in the next example.

6.2. "YearPredictionMSD" Dataset

In this section, we consider the "YearPredictionMSD" dataset, which is a subset of the Million Song Dataset. This dataset includes 515,345 songs, with 12 features of "timbre." We take these 12 features as well as 78 timbre covariances as predictors, i.e., 90 predictors in total. We take the year of release as the response and fit a linear model to this data set. We tried all four sampling methods on the dataset, and the summary of our results is shown in Figure 6.

The performance of the conditional squared bias of these four methods in this dataset has almost the same pattern as the performance in the Illumina dataset. Interestingly, in the middle panel, the graph shows that the conditional variance of LUVUNW is much better than all the other three methods, which also makes the MSE of LUVUNW decrease much faster than the other methods as the sample size increases. However, because of the large bias of LUVUNW, its best performance on MSE only shows up when the sample size is not too big compared to the entire dataset. The performance of BLEV and SLEV are still quite similar in this example, which is due to the lack of an extremely large leverage score in this dataset. As previously mentioned, if more influential points exist with leverage scores dominating the other data points, SLEV will be more robust than BLEV.

Figure 6. Empirical results for the "YearPredictionMSD" dataset; the notation is the same as that of Figure 5



7. SUMMARY

Sampling method, as an effective and general solution for big data problem, becomes more and more attractive. In this chapter, we focus on algorithm leveraging methods for solving large least-squares regression problems. It is a recently proposed popular sampling method, shown to be efficient in sampling influential data points. We compared the performance between Uniform Sampling and Basic Leverage Sampling, then discussed two newly-proposed leverage-based algorithms, Shrinkage Leverage Sampling Method (SLEV) and Leverage Unweighted Sampling Method (LEVUNW), aiming at minimizing MSE. Moreover, our case study provided a detailed evaluation of these algorithms on the real dataset. Based on the empirical results, we have shown that these two new algorithms, SLEV and LEVUNW, providing improved performance. However, there is no universal solution here. Based on the primary goal, careful consideration is needed before applying appropriate method. If the goal is to approximate, we suggest SLEV with either exact or approximate leverage scores. The reason is that SLEV results in much better conditional biases and variance compared to other existing methods according to empirical evidence. On the other hand, if our primary goal is to infer the true and most of the data does not have a relatively good linear pattern, or the sample size is much smaller than the entire data size, LEVUNW is recommended mainly due to its advantage in giving smaller variances. Finally, although not covered in this chapter, the leverage-based sampling method can also be applied to generalized linear models, time series models, variable selections, *etc*. A further refinement of the current methods and even brand new algorithms are under intensive development.

ACKNOWLEDGMENT

This research is partially supported by the NIH grants R01 GM113242 and R01 GM122080, NSF grants DMS-1228288, DMS-1438957, NSF ATD-1440037.

REFERENCES

ATLAS. (n.d.). *Trigger and Data Acquisition System*. Available: http://atlas.cern/discover/detector/trigger-daq

Avron, H., Maymounkov, P., & Toledo, S. (2010). Blendenpik: Supercharging LAPACKs least-squares solver. *SIAM Journal on Scientific Computing*, *32*(3), 1217–1236. doi:10.1137/090767911

Chen, C. P., & Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, *275*, 314–347. doi:10.1016/j.ins.2014.01.015

Chen, X., & Xie, M. G. (2014). A split-and-conquer approach for analysis of extraordinarily large data. *Statistica Sinica*, 1655–1684.

Drineas, P., Magdon-Ismail, M., Mahoney, M. W., & Woodruff, D. P. (2012). Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, *13*, 3475–3506.

Drineas, P., Mahoney, M. W., & Muthukrishnan, S. 2006. Sampling algorithms for l 2 regression and applications. *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, 1127-1136. doi:10.1145/1109557.1109682

Drineas, P., Mahoney, M. W., Muthukrishnan, S., & Sarl, S. (2011). Faster least squares approximation. *Numerische Mathematik*, *117*(2), 219–249. doi:10.1007/s00211-010-0331-6

Efron, B. (1979). Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, *7*(1), 1–26. doi:10.1214/aos/1176344552

Gai, K., & Li, S. 2012. Towards cloud computing: a literature review on cloud computing and its development trends. *2012 Fourth International Conference on Multimedia Information Networking and Security*, 142-146. doi:10.1109/MINES.2012.240

Gittens, A., & Mahoney, M. W. (2013). Revisiting the Nystrom method for improved large-scale mach*ine learn*ing. *ICML*, *28*(3), 567–575.

Golub, G. H., & Van Loan, C. F. (2012). *Matrix computations*. JHU Press.

Helwig, N. E., & Ma, P. (2016). *Smoothing spline ANOVA for super-large samples: scalable computation via rounding parameters.* arXiv preprint arXiv:1602.05208

Li, J., Jiang, H., & Wong, W. H. (2010). Modeling non-uniformity in short-read rates in RNA-Seq data. *Genome Biology*, *11*(5), 1–11. doi:10.1186/gb-2010-11-5-r50 PMID:20459815

Li, R., Lin, D. K., & Li, B. (2013). Statistical inference in massive data sets. *Applied Stochastic Models in Business and Industry*, *29*, 399–409.

Lin, N., & Xi, R. (2011). Aggregated estimating equation estimation. *Statistics and Its Interface*, *4*(1), 73–83. doi:10.4310/SII.2011.v4.n1.a8

Loeppky, J. L., Sacks, J., & Welch, W. J. (2009). Choosing the Sample Size of a Computer Experiment: A Practical Guide. *Technometrics*, *51*(4), 366–376. doi:10.1198/TECH.2009.08040

Ma, P., Mahoney, M., & Yu, B. (2014). A Statistical Perspective on Algorithmic Leveraging. *JMLR: Workshop and Conference Proceedings*, *32*, 91-99.

Ma, P., Mahoney, M. W., & Yu, B. (2015). A statistical perspective on algorithmic leveraging. *Journal of Machine Learning Research*, *16*, 861–911.

Ma, P., & Sun, X. (2015). Leveraging for big data regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, *7*(1), 70–76. doi:10.1002/wics.1324

Mahoney, M. W. (2011). Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning, 3*, 123-224.

Mahoney, M. W., & Drineas, P. (2009). CUR matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences of the United States of America*, *106*(3), 697–702. doi:10.1073/pnas.0803205106 PMID:19139392

Meng, X., Saunders, M. A., & Mahoney, M. W. (2014). LSRN: A parallel iterative solver for strongly over-or underdetermined systems. *SIAM Journal on Scientific Computing*, *36*(2), C95–C118. doi:10.1137/120866580 PMID:25419094

Mortazavi, A., Williams, B. A., Mccue, K., Schaeffer, L., & Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature Methods*, *5*(7), 621–628. doi:10.1038/nmeth.1226 PMID:18516045

Nagalakshmi, U., Wang, Z., Waern, K., Shou, C., Raha, D., Gerstein, M., & Snyder, M. (2008). The transcriptional landscape of the yeast genome defined by RNA sequencing. *Science*, *320*(5881), 1344–1349. doi:10.1126/science.1158441 PMID:18451266

nobelprize.org. (n.d.). *The Nobel Prize in Physics 2013*. Available: http://www.nobelprize.org/nobel_prizes/physics/laureates/2013/

Pratas, F., Trancoso, P., Sousa, L., Stamatakis, A., Shi, G., & Kindratenko, V. (2012). Fine-grain parallelism using multi-core, Cell/BE, and GPU Systems. *Parallel Computing*, *38*(8), 365–390. doi:10.1016/j.parco.2011.08.002

Scannicchio, D. (2010). ATLAS trigger and data acquisition: Capabilities and commissioning. *Nuclear Instruments & Methods in Physics Research. Section A, Accelerators, Spectrometers, Detectors and Associated Equipment*, *617*(1-3), 306–309. doi:10.1016/j.nima.2009.06.114

Shao, J., & Tu, D. (2012). *The jackknife and bootstrap*. Springer Science & Business Media.

top500.org. (2014). *June 2014*. Available at: https://www.top500.org/lists/2014/06/

Weisberg, S. (2005). *Applied linear regression*. John Wiley & Sons. doi:10.1002/0471704091

Wu, C.-F. J. (1986). Jackknife, bootstrap and other resampling methods in regression analysis. *the Annals of Statistics*, 1261-1295.

Xu, C., Zhang, Y., & Li, R. (2015). *On the Feasibility of Distributed Kernel Regression for Big Data.* arXiv preprint arXiv:1505.00869