Learning from Large-Scale Distributed Health Data: An Approximate Logistic Regression Approach

Che Ngufor CNGUFOR@MASONLIVE.GMU.EDU
School of Physics, Astronomy, and Computational Sciences, George Mason University, Fairfax VA 22030

Janusz Wojtusiak JWOJT@MLI.GMU.EDU
Department of Health Administration and Policy, George Mason University, Fairfax VA 22030

Abstract

Research in healthcare is increasingly depending on the access to and analysis of large distributed datasets. Coupled with the exponential rate at which data is being generated, the need for parallel processing is apparent if knowledge is to be efficiently extracted from these large data sets. The Hadoop-MapReduce framework has evolved into a popular platform for parallelization in many fields, including healthcare. Unfortunately, implementing iterative machine learning algorithms on Hadoop is inefficient, complex and time consuming.

To address this problem, this paper presents a computationally efficient method for computing the logistic regression. A simple approximation to the logistic function leads to a single-pass MapReduce algorithm. The method presented has two important virtues: computational simplicity and application. A series of numerical experiments on artificial and clinically relevant data sets showed that the method competes favorably with regards to accuracy, parallel efficiency and scalability compared to classical iterative methods.

1. Introduction

The healthcare system is comprised of multiple stakeholders including patients, providers, pharmaceutical companies, insurers, and government entities. Each of these groups generates pools of data on a daily basis. While such massive data is becoming very common in healthcare, analytic methods lagged behind in improving operational performance and adopting new technologies to handle the data the industry generates. The use of "Big data" in healthcare has the potential to revolutionize the way the various stakeholders operates and most importantly, the way patients are treated. "Big data" is described as high in volume, velocity and variety. In other words it refers to datasets so large and complex that it becomes difficult for conventional database management systems to capture, store and process efficiently. McKinsey Global Institute (Manyika et al., 2011) estimated that, Big data when used creatively and effectively by the U.S healthcare industry can help unlock more than $300 billion every year in additional value throughout the sector.

Over the years, researchers have investigated ways to reduce costs and improve efficiency and quality of healthcare. A popular approach has been to deploy decision support systems that combine machine learning, optimization and visualization techniques to assist in decision making. For example, machine learning assisted decision support systems helped providers reduce adverse drug reactions, reduce treatment error rates and hence liability claims, automatic discovering of drug treatment patterns in electronic medical records (EMRs), improve the diagnosis of rare diseases, predicting complications for patients undergoing various treatment options such as surgical procedures *etc.* However, these machine learning assisted decision support systems are designed and work efficiently for centralized systems with relatively small data sizes. Designing efficient machine learning assisted decision support systems for Big data systems is a very difficult task. The computational cost, storage and network bandwidth requirements may forbid the practical implementation of some learning algorithms.

Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013. JMLR: W&CP volume 28. Copyright 2013 by the author(s).

New technologies such as Hadoop[1] and MapReduce (Dean & Ghemawat, 2008) have been developed and optimized for efficient distributed computing. In this approach, machine learning problems are divided into multiple tasks, each of which is solved by one or more computers working in parallel. The benefits of such distributed systems in healthcare include higher performance at a lower cost, higher reliability and more scalability. Current trends show that more and more healthcare systems are adopting the Hadoop-MapReduce framework as the analytic platform of choice (Manyika et al., 2011; Villars et al., 2011).

The Hadoop and other distributed computing variants are greatly optimized for single-pass batch processing algorithms. However, most machine learning algorithms such as logistic regression (LR), support vector machines, neural networks, expectation maximization, k-means clustering *etc.*, are iterative or multi-pass algorithms. These algorithms load the same data repeatedly to improve and update learned parameters. The iterative application of MapReduce (MR) is known to be very costly and inefficient. To derive the most benefit from these technologies, there is need to design algorithms which are able to process data in a distributed fashion with minimal communication between the processes.

To address this problem, this work proposes a very simple and useful approximation to the maximum likelihood score equation of the LR. This easily leads to a "closed form" solution for the Maximum Likelihood Estimates (MLE) of the parameters. The closed form solution is shown to be very competitive with standard iterative methods and naturally leads to a cost-effective single-pass MR LR algorithm. Optionally, when high accuracy requirements are essential, the closed form solution can be used as starting points for iterative methods like the Newton-Raphson. This can reduce the convergence time to more than half. However, extensive experiments have been performed on both small and large real data sets and results from the closed form solutions alone competes rather favorably to iterative methods. The proposed single pass MR LR has two important virtues: computational simplicity and application. A series of numerical experiments on artificial and real medical data sets demonstrates the accuracy, parallel efficiency and scalability of the algorithm.

The rest of the paper is structured as follows: Section 2 discourses the potential benefits of the Hadoop-MapReduce framework in healthcare systems. Section 3 presents the LR model and its approximate close form solution while Section 4 discourses its implementation on MR. Numerical experiments are reported in Section 5 and Section 6 concludes the paper.

[1]http://hadoop.apache.org/

2. Distributed Healthcare Systems

The structure of healthcare data is very complex. It includes structured elements (coded using one of many available standards or using local terminologies), and unstructured elements such as text, images, videos, and recorded speech. In a sense, some data elements are in spirit similar to tweets, Facebook postings or LinkedIn profiles. Therefore it is reasonable to expect that the technologies and environments powering these applications should be able to do the same with healthcare data. While massive data is continuously being generated in the healthcare, the industry has lagged behind in improving operational performance by adopting these new technologies that at present have developed rapidly. However, current trends are showing rapid adoption of these technologies in healthcare. This will bring in many benefits but at the same time new challenges.

The Appache Hadoop Distributed file system (HDFS) is an open source distributed file system on which several applications can be run on large clusters of commodity hardware. Its development was inspired by Google's MapReduce and Google File System. In HDFS, the data is broken into blocks and spread throughout the cluster. MR tasks can then be carried out on the smaller subsets of data thus accomplishing scalability. For Big data healthcare systems, the Hadoop-MapReduce framework is uniquely capable of storing a wide range of healthcare data types including electronic medical records, genomic data, financial and claims data *etc.*, and offers high scalability, reliability and availability than traditional DBMS. In addition, intelligent functional modules such as specialized machine learning algorithms for image analysis and recognition, diagnosis, surveillance, detection, notification *etc.*, can be built on it.

A typical application of Hadoop and machine learning to distributed healthcare data can be illustrated by the following example. To better improve treatment outcomes for its patients, a hospital system wants to explore why patients choose a specific treatment option. In addition, the hospital wants to examine treatment decisions with respect to quality of life post treatment. With Hadoop, MR and LR, the hospital can analyze a large distributed patient clinical, demographic and physician database that spans multiple hospitals (assuming no centralized EMR is available) to identify factors influencing treatment choices. Prognoses for treatment options and complications can then be made. Automatic follow up inquiries or notifications can also be sent out to patients. Treatment decision regret can be determined from responses of patients who were predicted false positive by the model.

This example illustrates how machine learning built on top of the Hadoop framework can support human decision making in complex situations involving multiple factors. More generally, it illustrates the fact that, it is only through the design of highly scalable analytic methods can discoveries be made that justifies the collection of large healthcare data.

One drawback of the Hadoop-MapReduce framework however is that, the HDFS storage strategy is not suitable for high computational complexity applications which are typical of most machine learning algorithms. HDFS is optimized for single-pass batch processing algorithms such as summary statistics (counts, mean and standard deviations), specialized linear discriminant analysis and feature extraction *etc.* Most machine learning algorithms are iterative, they process data iteratively to update and improve learned parameters values. These algorithms are very costly and inefficient in terms of computation and time to run on MR. This can be a serious drawback for healthcare systems where lives are at stake. The next Section will describe a modification to the LR learning algorithm for efficient MR implementation.

3. Approximate MLE for LR Model

Logistic Regression (LR) is a learning algorithm with sound statistical background and widely used in machine learning, data mining, and statistics. It has found widespread use in the biomedical and epidemiological research. The popularity of LR in healthcare research can be attributed to its simplicity and interpretability of model parameters. Typical use of LR in healthcare may include: assessing the presence or absence of a disease, predicting survival of breast cancer, determining how likely people may utilize the emergency room rather than going to a primary physician for care, how likely women are to perform monthly breast self-exam *etc.* Many factors may or may not influence these outcomes. Previous studies, published results, medical tests, records and charts of other patients may be helpful to build a risk factor database of the disease or the outcome of interest.

Like any learning method, LR tries to find the optimal model that explains the relationship between a binary outcome or dependent variable and a set of independent variables. The next two sections will briefly present the LR model and its parameter estimation. A full treatment can be found in standard statistics text like Hastie et al. (2001).

3.1. The LR Model

Let $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ be a set of training examples where the random variables $Y_i = y_i \in \{0, 1\}$ are binary and $X_i = x_i \in \mathbb{R}^p$ are d-dimensional feature vectors.

There is considerable empirical evidence that the probability of occurrence of a binary response variable $y \in \{0, 1\}$ taking the value $y = 1$ in a LR depending on a number of independent variables $x = (x_0, x_1, \ldots, x_d)$ is given by the logistic function

$$\mu(\beta) = p(y = 1|x; \beta) = \frac{1}{1 + e^{-x^T\beta}}$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_d)$ is the vector of parameters. The vector x has been augmented with $x_0 = 1$ to account for the intercept term. Assuming the observations are independent and that each y_i is a Bernoulli random variable, the log-likelihood function is given by

$$l(\beta) = \beta^T X^T Y - \sum_{i=1}^{n} \log\left(1 + e^{x_i^T\beta}\right) \tag{1}$$

where Y is the $n \times 1$ response vector and X is an $n \times (d+1)$ matrix also called the model matrix.

Typically, the method of maximum likelihood is used to estimate the unknown parameters β. The log-likelihood equation is differentiated with respect to β, set to zero and solve

$$\frac{\partial l(\beta)}{\partial \beta} = X^T Y - \sum_{i=1}^{n} \mu_i x_i = X^T(Y - \boldsymbol{\mu}(\beta)) = \mathbf{0} \tag{2}$$

where $\boldsymbol{\mu}(\beta) = (\mu_1(\beta), \mu_2(\beta), \ldots, \mu_n(\beta))$. Unfortunately, Eq. 2 is nonlinear and there is no close form solution. The traditional approach is to approximately solve it using iterative methods.

3.2. Iterative Methods

Let $S(\beta) = X^T(Y - \boldsymbol{\mu}(\beta))$, finding a solution to the LR score equations is equivalent to finding the zeros of the non-linear score function $S(\beta)$. There are many numerical optimization methods to do this, the Newton-Raphson method is perhaps one of the most popular methods. The Newton's method takes the first degree Taylor series approximation of $S(\beta)$ at a point β^{old} (initial guess), set it to zero and solve for a new approximate solution β^{new}. The update process is repeated until convergence. Specifically, starting with β^{old}, a single Newton update is

$$\beta^{new} \longleftarrow \beta^{old} + \left(X^T W X\right)^{-1} X^T \left(Y - \boldsymbol{\mu}\left(\beta^{old}\right)\right) \tag{3}$$

where W is a $n \times n$ diagonal matrix of weights having $\mu_i(\beta)(1-\mu_i(\beta))$ as the $i'th$ diagonal element. The matrix X^TWX is the second-derivative of the score function also called the Hessian matrix.

In general the Newton method will converge to the unique maximum though over-fitting may occur. In particular, the LR classifier tends to over-fit when the dimension is high. This problem is usually solved by regularization i.e imposing a penalty on the magnitude of the parameter values. This penalty should be minimized while simultaneously maximizing the likelihood. The L_2 regularization is obtained by subtracting λ times the squared L_2 norm of the parameter vector from the log-likelihood i.e

$$l(\beta) = \beta^T X^T Y - \sum_{i=1}^{n} \log\left(1 + e^{x_i^T \beta}\right) - \lambda \|\beta\|^2$$

λ reflects the strength of regularization.

Computing large scale matrices and their inverses can be computationally very expensive, for this reason many authors have resorted to the Hadoop platform to solve the problem through the MR framework (Chu et al., 2007; Teo et al., 2010; Liu & Liu, 2011; Tamano et al., 2011). However it might still be computationally demanding to compute the Hessian on Hadoop. Gradient based methods that avoid computing the Hessian are more preferable. The batch gradient descent (GD) and stochastic gradient descent (SGD) L_2 regularized methods are given respectively by

$$\beta^{new} = \beta^{old} + \frac{\alpha}{n}\left[X^T\left(Y - \boldsymbol{\mu}\left(\beta^{old}\right)\right) - 2\lambda\beta^{old}\right] \quad (4)$$

$$\beta^{new} = \beta^{old} + \alpha\left[\left(y_i - \mu_i\left(\beta^{old}\right)\right)x_i - 2\lambda\beta^{old}\right] \quad (5)$$

where the scalar $\alpha > 0$ is called the learning rate, and can be chosen as fixed, adaptive or according to a fixed decreasing schedule. Note that $\lambda = 0$ corresponds to unregularized gradient methods. In SGD, instead of averaging the gradient over the training set, each iteration consist of choosing an example (x_i, y_i) at random from the training set and updating the parameter β. This economizes on the computational cost at every iteration. Several passes over the data are made until the algorithm converges.

3.3. Taylor Approximation of the LR Function

The Newton-Raphson method and its variants obtain approximate solution to the LR score equation through a first order Taylor series approximation of the score function. This paper proposes a different approach. Instead of directly taking a Taylor expansion of the score function, the first order Taylor series approximation of the logistic function is used in the score function. The resulting equation is just the least squares normal equations with a closed form solution for the MLE of β.

The Taylor series expansion of the logistic function $f(z) = 1/(1+e^{-z})$ around $z = 0$ is given by

$$f(z) = \frac{1}{2} + \frac{1}{4}z - \frac{1}{48}z^3 + \frac{1}{480}z^5 - \frac{17}{80640}z^7 + \ldots$$

For the LR, the decision boundary separating the two classes is given by $x^T\beta = 0$. Data points close or on the decision boundary are more apt to be misclassified and hence very important. The LR learning can thus be restricted to learning the characteristics of these points. Taking the terms linear in $x^T\beta$ in the Taylor expansion of $\mu(\beta)$ around $x^T\beta = 0$ and substitute in the score equation gives

$$X^T(Y - \boldsymbol{\mu}(\beta)) \approx X^T\left(Y - \frac{1}{2}I_n - \frac{1}{4}X\beta\right) = \mathbf{0} \quad (6)$$

where I_n is a $n \times 1$ vector of ones. The solution to this "normal equation" is the "approximate MLE" of β given by

$$\hat{\beta} = \left(\frac{1}{4}X^TX\right)^{-1} X^T\left(Y - \frac{1}{2}I_n\right) \quad (7)$$

The theoretical guarantee of the approximation quality is given by the error of the Taylor series and is expressed in terms of the degree of the expansion.

It can be easily shown that the corresponding L_2 regularized approximate MLE is given by

$$\hat{\beta} = \left(\frac{1}{4}X^TX + 2\lambda I_{d+1}\right)^{-1} X^T\left(Y - \frac{1}{2}I_n\right) \quad (8)$$

where I_{d+1} is a $(d+1) \times (d+1)$ identity matrix with the first entry corresponding to the intercept term set to zero.

4. Single-Pass and Multi-Pass Algorithms on MapReduce

The MR framework is based on a typical divide and conquer parallel computing strategy. Any application that can be designed as a divide and conquer application can generally be set-up as a MR program. The application core of MR consists of two functions: a *Map* and a *Reduce* function. The input to the Map function is list of *key-value* pairs. Each key-value pair is processed separately by each Map function and outputs a key-value pair. The output from each Map is then shuffled so that values corresponding to the same key are grouped together. The Reduce function aggregates the list of values corresponding to the same key based on the user specified aggregating function. Typically all that is required is for the user to provide the Map and Reduce functions. Data partitioning, distribution, replication, communication, synchronization and fault tolerance is handled by the Hadoop platform.

The Hadoop-MapReduce framework is designed for embarrassingly parallel data intensive tasks and turns to perform poorly for complex and iterative or *multi-pass* applications. For example, the naive implementation of the GD method given by Eq. 4 on MR will have each Map function computes the value of $X^T\left(Y - \boldsymbol{\mu}\left(\beta\right)\right)$ using β computed at the previous iteration. The Reducer then sums up the output from the mappers and update β for the next iteration. This process requires multiple MR jobs to be executed for every iteration and increases with the number of iteration required for convergence making the whole process very inefficient. Unfortunately, this expensive multi-pass implementation of the gradient method is common even in recent distributed machine learning literature (Chu et al., 2007; Liu & Liu, 2011; Tamano et al., 2011).

On the other hand, non-iterative or *single-pass* machine learning algorithms are readily amenable to the MR framework. To this end, the computation of the presented approximate MLE (ApproxMLE) of the LR on MR is simple and straightforward. Specifically, it is a single-pass solution of Eq. 8 which is a simple linear equation of the form $A\beta = b$. Each Map function computes its contribution to the matrix A and the vector b. The reducer then aggregates all these contributions and solves for $\beta = A^{-1}b$. The matrix A is $(d+1) \times (d+1)$, so for high dimensional data sets it can be computationally expensive to compute the inverse. Fortunately, there exist efficient parallel approaches for solving linear systems. Two interesting recent algorithms for linear systems on MR are worth mentioning here: Jakovits et al. (2011) Monte Carlo approach and Fliege (2012) randomized algorithm.

Two papers recently implemented single-pass versions of the GD and SGD on MR. In Mann et al. (2009), the full GD is solved for the parameter vector by each mapper. The reducer simple averaged the solutions to obtain a global solution. This approach will be called Weighted GD (WGD). Zinkevich et al. (2010) used Mann et al. (2009) idea but replaces the GD with the SGD. Similarly this approach will be called Weighted SGD (WSGD). Both methods guarantee convergence of the parameter distribution; however the averaging may raise some questions. Equal weights averaging assume all mappers contribute equally to the parameter estimate. To verify if averaging has any impact on the solution, a weighted version of ApproxMLE (W.ApproxMLE) is also implemented.

5. Experiments

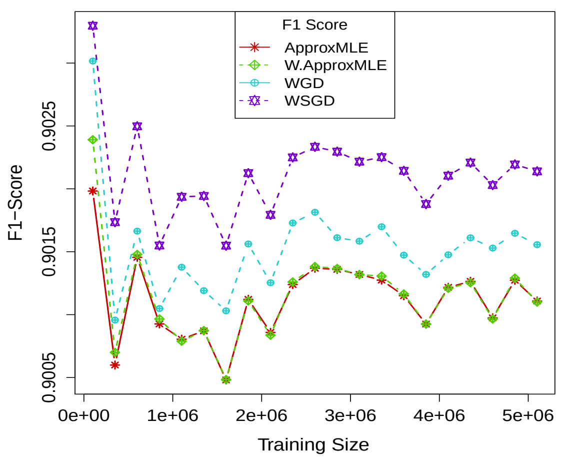
A series of experiments on artificial and real medical data sets are performed in this section to demonstrate the accuracy, parallel efficiency and scalability of the proposed method. The experiments are performed using Hadoop version 1.0.4 on a cluster of three machines. One of the machines is an 8 core, 16 GB RAM and 1 TB storage. The other two are a 6 core, 8 GB RAM and 1 TB storage.

The learning rate α for WGD is set to $\alpha = 0.5$ and to $\alpha = 1/n$ for WSG where n is the sample size of the data block assigned to a mapper. All experiments used the same regularization parameter $\lambda = 0.05$. The performances of the algorithms are assessed and compared using a number of performance measures described next.

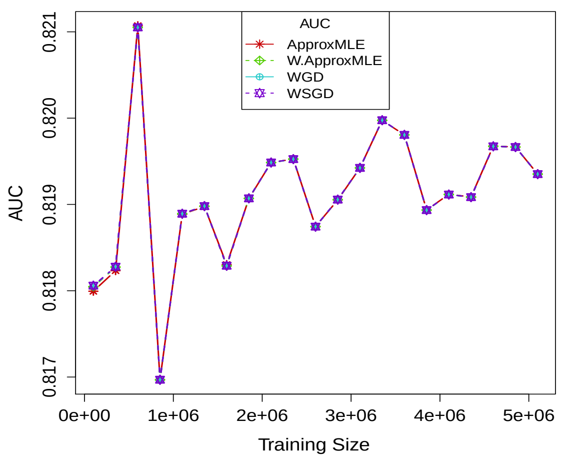
5.1. Performance Measures

Many real-world classification data sets and especially healthcare data are often highly imbalanced with one class significantly underrepresented. Performance measures such as the classification accuracy and error rate have been shown in many studies to be inappropriate to evaluate the performance of a classifier on imbalanced data sets. A trivial classifier that predicts every example as majority class can still achieve a very high accuracy. As a result, this can be very problematic when comparing the performance of different learning algorithms across different data sets. A performance measure that is not affected by the class distribution of the data set is therefore preferable. In lieu of the accuracy, performance measures such as the *precision, recall, F1-Score, area under the receiver operating characteristic curve (AUC)* and the *G-mean* have been adopted by the machine learning research community as preferable measures for imbalanced learning problems (He & Garcia, 2009).

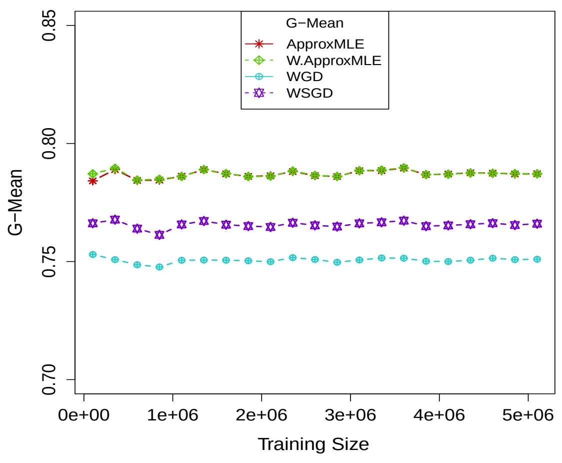
The *F1-Score* is commonly taken as the harmonic mean between the *precision* and *recall* however; it is still sensitive to the data distribution. *G-mean* is defined as the geometric mean of the classifier accuracies observed separately on the minority and majority classes and is insensitive to class distribution.



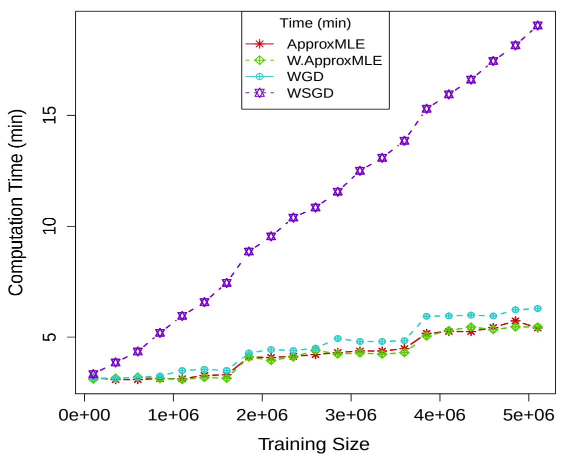
(a) F1-Score



(b) AUC



(c) G-Mean



(d) Computation Time

Figure 1. Performance vs Training Set Size

5.2. Synthetic Data

A simple two population multivariate normal distribution with equal covariance matrix $\mathcal{N}_d(\mu_i, \Sigma)$ is generated. The mean vectors are set to $\mu_1 = (0, 0, \dots, 0)$ and $\mu_2 = \mu_1 + 0.5$. $\Sigma = \mathbf{diag}(1.5, \dots, 1.5)$ is a diagonal matrix with 1.5 on the diagonal.

With the proportion of observations falling in one population set to 0.2, a series of data sets of sizes 100×10^3 to 5100×10^3 in increments of 250×10^3 is generated. This gives a total of 21 MR experiments. For each data set, 10% is reserved for testing and the rest for training. All algorithms are trained and tested on the same training and testing data sets.

Figures 1 shows the performance of the four algorithms on the test set plotted against the training set sizes. It can be seen that the performance of the two Approximate MLE methods are identical on all experiments. This result shows that averaging of MR has no effect on the final solution. With respect to the *F1-Score* (Figure 1 (a)), the performances of the two gradient methods are superior to the approximate methods. But as noted earlier, the *F1-Score* is sensitive to class distribution that varies across different data sets. All algorithms have identical performance with regards to *AUC* (Figure 1 (b)). For the *G-mean*, the approximate MLE methods significantly outperformed the gradient methods. Zinkevich et al. (2010) reported linear scalability of WSGD on MapReduce. Figure 1 (d) shows that WSGD scales linearly with time. For the problem sizes investigated in this section, the ApproxMLE, W.ApproxMLE and WGD significantly scales better than WSGD. However, these are relatively small data sets compared to current data sets generated in the healthcare.

5.3. Real Medical Data

This section presents the performances of the ApproxMLE, W.ApproxMLE, WGD and WSGD algorithms on two real medical data sets.

5.3.1. Breast Cancer Data

The first data set consist of 2,392,998 screening mammograms from women included in the Breast Cancer Surveillance Consortium (BSCS) (Ballard-Barbash et al., 1997) registries. All women in the study did not have a previous diagnosis of breast cancer and did not have any breast imaging in the nine months preceding the screening mammogram. However, all women had undergone previous breast mammography in the prior five years (though not in the last nine months). Cancer outcomes for each mammography record in BCSC are obtained by linking with cancer outcomes records from cancer registries such as the National Cancer Institute's Surveillance, Epidemiology, and End Results (SEER).

The data contained a total of thirteen predictors and an independent variable indicating diagnosis of breast cancer within one year of the screening mammogram. A total of 11638 women were diagnosed with breast cancer within one year of screening. The data also contained an indicator variable for cases included in the training or validation data set. Using this variable, the final training set consisted of 75% of the data. Barlow et al. (2006) previously used this data to build LR cancer risk predictive models for two cohort groups: premenopausal and postmenopausal women. However, in this study the focus is on the predictive power, parallel efficiency and scalability of the methods and not on identifying cancer risk factors. Thus all variables were used for model building.

5.3.2. Record Linkage Comparison Patterns Data Set

Record linkage is the task of identifying record in a data set that refers to the same record across different data sources. Record linkage is commonly applied to biomedical databases such as disease registries or more generally when data about one patient needs to be linked with the patient data from other sources. For example the breast cancer data described in Section 5.3.1 was obtained by linking BSCS records with SEER records. Another very popular example is the SEER-Meicare[2] linked database, created by linking SEER records to the Medicare beneficiaries database.

The primary goal of record linkage is to remove or reduce false-match errors i.e when records belonging to different patients are wrongly classified as equal. Duplicate entries in cancer registries can lead to incorrect cancer incidence estimation. In some applications it may also be important to minimize the false non-match error i.e when one patient record is linked to multiple records. The first step in record linkage is to create record pairs and then transforming them to comparison patterns. A classification model can then be constructed to predict matches based on the patterns.

The record linkage comparison data set used in this study is from the UCI machine learning repository (Frank & Asuncion, 2010; Sariyar et al., 2011) and consist of 5,749,132 record pairs compiled at the Epidemiological Cancer Registry of the German state of North Rhine-Westphalia (Epidemiologisches Krebsregister[3]). The comparison patterns were classified as "match" or "non-match" of which 20,931 were matches.

Table 1 shows the performance of the four LR algorithms on the breast cancer and record linkage data. All algorithms show excellent performances. The significant difference is in the execution time. Once more, the approximate MLE methods have similar computation time that is significantly less than that of the gradient methods. For the record linkage data set the average execution time of ApproxMLE and W.ApproxMLE is about 33 times less than the time taken by WGD and 8 times less than that of WSGD.

Judging from the performances on the synthetic and real data sets, the time scalability of WGD is unclear however, that of WSGD remains linear.

6. Conclusion

This paper presented a simple and useful approximation to the Logistic Regression score function leading to a closed form solution for the Maximum Likelihood Estimates thus avoiding the need for iterative methods. The approximate MLE computation is directly amenable to the MR framework as a single-pass algorithm. A series of numerical experiments on both synthetic and real medical data demonstrates that the

[2]http://healthservices.cancer.gov/seermedicare/

[3]http://www.krebsregister.nrw.de

Table 1. Performance on Breast Cancer and Record Linkage data sets

Data	Algorithm	*F1-Score*	*AUC*	*G-Mean*	Time (min)
Breast Cancer	ApproxMLE	1.00	0.92	1.00	3.24
	W.ApproxMLE	1.00	0.90	1.00	3.18
	WGD	1.00	0.89	1.00	38.63
	WSGD	1.00	0.90	1.00	13.81
Record Linkage	ApproxMLE	1.00	1.00	0.79	3.18
	W.ApproxMLE	1.00	1.00	0.79	3.28
	WGD	1.00	1.00	0.79	105.52
	WSGD	1.00	1.00	0.79	25.61

proposed method competes favorable with single-pass versions of iterative gradient methods on MR. The method has excellent performance in terms of accuracy, parallel efficiency and time scalability.

The paper has also presented cases in which the algorithm is particularly suitable for healthcare applications, and analysis of large-scale clinical/administrative data to improve patient outcomes. There are several potential future directions of the presented work. Immediate plans include: error and sensitivity analysis, more detailed testing on real datasets and how to apply the algorithm to data standardization and coding often present in healthcare.

References

Ballard-Barbash, Rachel, Taplin, Stephen H, Yankaskas, Bonnie C, Ernster, Virginia L, et al. Breast cancer surveillance consortium: a national mammography screening and outcomes database. *AJR. American journal of roentgenology*, 169(4): 1001–1008, 1997.

Barlow, William E, White, Emily, Ballard-Barbash, Rachel, Vacek, Pamela M, et al. Prospective breast cancer risk prediction model for women undergoing screening mammography. *Journal of the National Cancer Institute*, 98(17):1204–1214, 2006.

Chu, Cheng, Kim, Sang Kyun, Lin, Yi-An, Yu, YuanYuan, Bradski, Gary, Ng, Andrew Y, and Olukotun, Kunle. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

Dean, Jeffrey and Ghemawat, Sanjay. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

Fliege, J. A randomized parallel algorithm with run time $\mathcal{O}(n^2)$ for solving an $n \times n$ system of linear equations. *ArXiv e-prints*, September 2012.

Frank, Andrew and Asuncion, Arthur. Uci machine learning repository. 2010.

Hastie, Trevor, Tibshirani, Robert, and Friedman, J. H. *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. New York: Springer-Verlag, 2001.

He, Haibo and Garcia, Edwardo A. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

Jakovits, Pelle, Kromonov, Ilja, and Srirama, Satish Narayana. Monte carlo linear system solver using mapreduce. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pp. 293–299. IEEE, 2011.

Liu, Zhen and Liu, Meng. Logistic regression parameter estimation based on parallel matrix computation. *Theoretical and Mathematical Foundations of Computer Science*, pp. 268–275, 2011.

Mann, Gideon, McDonald, Ryan, Mohri, Mehryar, Silberman, Nathan, and Walker, Dan. Efficient large-scale distributed training of conditional maximum entropy models. *Advances in Neural Information Processing Systems*, 22:1231–1239, 2009.

Manyika, James, Chui, Michael, Brown, Brad, Bughin, Jacques, Dobbs, Richard, Roxburgh, Charles, and Byers, Angela Hung. Big data: The next frontier for innovation, competition, and productivity. *McKinsey Global Institute*, pp. 1–137, 2011.

Sariyar, Murat, Borg, Andreas, and Pommerening, Klaus. Controlling false match rates in record linkage using extreme value theory. *Journal of biomedical informatics*, 44(4):648–654, 2011.

Tamano, Hiroshi, Nakadai, Shinji, and Araki, Takuya. Optimizing multiple machine learning jobs on mapreduce. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pp. 59–66. IEEE, 2011.

Teo, Choon Hui, Vishwanthan, SVN, Smola, Alex J, and Le, Quoc V. Bundle methods for regularized risk minimization. *The Journal of Machine Learning Research*, 11:311–365, 2010.

Villars, Richard L, Olofson, Carl W, and Eastwood, Matthew. Big data: What it is and why you should care. *White Paper, IDC*, 2011.

Zinkevich, Martin, Weimer, Markus, Smola, Alex, and Li, Lihong. Parallelized stochastic gradient descent. *Advances in Neural Information Processing Systems*, 23(23):1–9, 2010.