USING A SCALABLE FEATURE SELECTION APPROACH FOR BIG DATA REGRESSIONS

by

Qingdong Cheng

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

August 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Baijian Yang, Chair

Department of Computer and Information Technology

Dr. Dominic Kao

Department of Computer and Information Technology

Dr. Tonglin Zhang

Department of Statistics

Approved by:

Dr. Eric T. Matson

Head of the Graduate Program

Dedicated to my grandfather.

ACKNOWLEDGMENTS

Firstly, I would like to thank my wife Cong for her love, support, and encouragement. I have been interested in computer science since I was in high school. Without the encouragement of my wife, I would not have the courage to pursuit the master's degree in computer related area. Moreover, I would like to thank my family for their support. Especially, I would like to thank my son Michael. His birth encouraged me to apply the master's program. Also, I would like to sincerely thank my advisor Professor Baijian Yang for helping me start this master's program. I also would like to thank Professor Yang for his continuous support and instruction throughout my master's study and the research of the thesis. Last, but not least, my thanks also go to the members of my major committee, Tonglin Zhang, and Dominic Kao for providing their insightful comments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
HDFS	Hadoop Distributed File System
I/O	Input/Output
MB	Megabyte
OS	Operating System
RDD	Resilient Distributed Datasets
WSS	Working Sufficient Statistics
YB	Yottabyte
ZB	Zettabyte

GLOSSARY

Apache Spark – Apache Spark an in-memory open-source distributed processing and analysis framework for solving big data and machine learning problems (Meng et al., 2016).

Feature Selection – The process of excluding irrelevant variables from a machine learning model (James, Witten, Hastie, & Tibshirani, 2013).

Logistic Regression – A type of regression analysis that is often utilized to describe the relationship among a binary dependent variable and one or multiple independent variables (Morgan & Teachman, 1988).

Regression – Regression analysis is the statistical process that estimate the model among variables, usually, one dependent variable and multiple independent variables (Yan & Su, 2009).

ABSTRACT

Author: Cheng, Qingdong. M.S.
Institution: Purdue University
Degree Received: August 2019
Title: Using a Scalable Feature Selection Approach For Big Data Regressions
Major Professor: Baijian Yang

Logistic regression is a widely used statistical method in data analysis and machine learning. When the capacity of data is large, it is time-consuming and even infeasible to perform big data machine learning using the traditional approach. Therefore, it is crucial to come up with an efficient way to evaluate feature combinations and update learning models. With the approach proposed by Yang, Wang, Xu, and Zhang (2018), a system can be represented using small enough matrices, which can be hosted in memory. These working sufficient statistics matrices can be applied in updating models in logistic regression. This study applies the working sufficient statistics approach in logistic regression machine learning to examine how this new method improves the performance. This study investigated the difference between the performance of this new working sufficient statistics approach and performance of the traditional approach on Spark's machine learning package. The experiments showed that the working sufficient statistics method could improve the performance of training the logistic regression models when the input size was large.

CHAPTER 1. INTRODUCTION

This chapter introduces the relevant background, the research question, the scope, and the significance of the research problem. This chapter also covers the scope, assumptions, limitations, and delimitations of the study.

1.1 Background

Machine learning is an important discipline in the area of artificial intelligence. Machine learning algorithms have great practical value in academia and industry. Recently, machine learning has been successfully applied to different fields (Angelova, Krizhevsky, & Vanhoucke, 2015; Ba, Mnih, & Kavukcuoglu, 2014; Frome et al., 2013). However, with the explosive growth of data all over world nowadays, data has been accumulated at an unprecedented rate. Approximately, Walmart creates about 2.5 petabytes of data hourly (McAfee, Brynjolfsson, Davenport, Patil, & Barton, 2012). The term big data was frequently mentioned in various contexts. By 2020, the volume of the data on earth is expected to reach 44 zettabytes (Zwolenski, Weatherill, et al., 2014). According to Mills et al. (2012), the exponential growth of data will keep going in the future. It is an era of data deluge today. There are various distributed file systems and distributed processing frameworks to solve big data problems, including Storm, Hadoop, and Spark. However, the scenarios of big data make machine learning much more challenging than before because it is infeasible to load the data into a system with limited memory (Yang & Zhang, 2016).

In many fields, such as data mining and document classification, data sets are large in size, which contain many attributes and records. Therefore, it is inefficient to process the whole dataset. A big challenge in big data analysis is to handle high-dimensional, sparse data. For instance, network traffic and large-scale social networks generate high-dimensional data (Kolda & Sun, 2008). Researchers can eliminate the irrelevant attributes through attribute selection and increase the effectiveness of the analysis task, thus improving the accuracy of the model and reducing the running time. Furthermore, as

Yang and Zhang (2016) pointed out, no single model fits all scenarios. To evaluate a model, researchers need to consider many aspects, such as feature selection, training data and testing data split, and validation. All these imply that the learning process will scan the data entire sets several times.

1.2 Problem Statement

The traditional approaches of machine learning processing need to go through the data multiple times. Also, many research questions come from domains with hundreds to thousands of features (Guyon & Elisseeff, 2003). When the data volume is large, it is not efficient to perform machine learning tasks with all the features. It is necessary to evaluate models with different features, due to the following reasons.

- The performance of some learning algorithms will become worse because of irrelevant or redundant features (Weston et al., 2001).
- The irrelevant features will increase the training samples size.
- Irrelevant features will decrease computational efficiency (Lin, Zhang, & Hung, 2014).

Therefore, it is beneficial to select appropriate features before performing machine learning model training. Yang and Zhang (2016) have pointed out that the working sufficient statistics matrices can be used to train logistic regression models. Therefore, it can overcome the memory barrier when the input size is large. A dataset with large size can be represented using these relatively small matrices. This study aims to utilize the matrices of working sufficient statistics on logistic regression in Spark to examine if and how it will improve the performance.

1.3 Scope of Problem

There are different big data processing platforms, including Apache Hadoop, Apache Spark, Apache Storm, Samza, etc. (Zhao, Chandrashekar, Lee, & Medhi, 2015). Instead of investigating other distributed computing frameworks, this study will mainly focus on Apache Spark and only explore how to utilize the approach of arrays of working sufficient statistics on Apache Spark. Spark is a cluster computing platform for big data analysis (Zaharia, Chowdhury, Franklin, Shenker, & Stoica, 2010). If the working sufficient statistics approach can help to improve the performance of logistic regression on Spark, further research would take other distributed processing frameworks into account.

1.4 Significance of the Problem

In recent years, numerous fields, such as optical observations, optical monitoring, healthcare, sensors, user data, Internet, and supply chain systems, have produced massive amounts of data in the past 20 years. The more appropriate description may be infinite data. Although technology captures and stores enormous quantities of data, one of the huge challenges is to handle such enormous amounts of data efficiently (Witten, Frank, Hall, & Pal, 2016). To extract trends from the data and transfer data into information and knowledge are crucial tasks. When applying machine learning models on these datasets, it is significant to find an efficient approach to identify appropriate features and accurate learning models. The conventional way of machine learning processing involves frequent I/O operations, such as the Hadoop framework, which is extremely time-consuming. Apache Spark utilizes RDDs, and it improves efficiency compared to the Hadoop framework. However, since the memory is a limited resource and the size of data is growing, the conventional logistic regression training approach will become inefficient. If the working sufficient statistics method can break the memory barrier when training the logistic regression model with large datasets, big data researchers and practitioners can perform big data analysis and machine learning on large datasets with limited memory.

1.5 Statement of Purpose

This research examines if the new approach based on the arrays of working sufficient statistics can improve the performance of the learning process of logistic regression for big data problems. The study will utilize this new approach and compare its performance with that of the traditional approach on Spark.

1.6 Research Questions

This study investigates and answers the research questions that if the working sufficient statistics approach could be applied in selecting features and training logistic regression models, and if this new approach can outperform the existing API in Spark. In other words, this study is designed to answer the following questions:

1. Could the use of working sufficient statistics improve the efficiency of logistic regression on Spark?
2. Could the use of working sufficient statistics reduce the memory usage of logistic regression on Spark?

1.7 Assumptions

The assumptions of the study are the following:

- In the scenario of big data, the memory is too small to load the whole dataset.
- In order to obtain an accurate model, multiple models are trained in a machine learning process.
- The dataset is appropriate for the logistic regression model.

1.8 Limitations

This study includes the following limitations.

- The cluster is running on the Spark cluster that has three machines.
- There are some missing values in the dataset.
- There are some outliers in the dataset, which may make the logistic model unable to converge.
- The dataset has been normalized using the z-score method. In some features, the variance of the data is still large.
- The memory of the machines in Spark cluster that was used this study is limited.
- Simulated data is used in experiment 2.

1.9 Delimitations

The delimitations of the study are:

- The version of Spark used in this study is 2.3.2.
- The Spark jobs are all implemented in Scala.
- The Scala version is 2.11.8.
- The Java JDK version is 1.8.0_191.
- There are two worker nodes and one master node in the Spark cluster.

1.10 Summary

This chapter focused on the discussion of the background, the research problem, the significance of the research question, the limitations, and so on. In the next chapter, the recent work of other researchers on big data analysis and feature selection will be discussed.

CHAPTER 2. REVIEW OF LITERATURE

In this chapter, a literature review on big data, data analysis methods, and data analysis tools is provided. This chapter also offers a review of recent work on feature selection.

2.1 Overview of Big Data

Data is considered as a crucial resource in the information age. Especially, big data is important in industry. In industry, the demand for data scientists is increasing. In the past 20 years, data has been growing faster and faster. According to International Data Corporation IDC (Gantz & Reinsel, 2011), data generated and copied in 2011 exceeds 1.8 ZB, which is nine times the data growth in the past five years and will increase at a rate of doubling every two years. Therefore, the field of big data research has captured interests from both industries and academia (Dubey, Gunasekaran, Childe, Wamba, & Papadopoulos, 2016). For example, reports from industry and public media, such as Economists, New York Times, US National Public Radio, are flooded with relevant information on big data. Government departments set up major projects to accelerate the development of big data. Therefore, the era of big data has come (Gupta & Rani, 2018). Before the era of big data, people work hard to extract information and draw a conclusion based on small data (James et al., 2013). The big data can offer more in-depth insights and broader scale understandings (Sagiroglu & Sinanc, 2013). People in various fields are making informed decisions by analyzing big data. If you can effectively manage big data, you can bring tremendous changes to many areas, such as science and environmental modeling, health care, and energy protection. The research report of McKinsey shows that the potential value of global personal location information data has reached 700 billion, and it can reduce the cost of product development and assembly by more than 50% (Manyika et al., 2011).

2.1.1 Definition of Big Data

The definition of big data presents a trend of diversification, and it is even more difficult to reach consensus. Essentially, big data means large volumes of data. Also, big data has other characteristics. Many pieces of literature defined big data. The following definitions are important.

In the report of IDC in 2011, big data was defined like this (Gantz & Reinsel, 2011): big data technology extracts meaningful information from large-scale data. There are four notable features defined in this definition, namely volume, variety, velocity, and veracity. The use of the "4Vs" definition is also extensive. The volume of data generated every day is massive. Every day, roughly 2.5 quintillion bytes of data were generated (Ganz, Barnaghi, & Carrez, 2013). Variety means that the data sources are various. Wearable devices, mobile phones, tablets, and healthcare devices, etc. are all possible data points. The velocity is the measure of the rate of the data generated in various sources. This unique characteristic is not only being limited to how fast the data coming in but also how fast the data flow. For example, a regular car can have more than 100 sensors. All these sensors are generating data constantly. Data is generated and stored in databases every second. Therefore, the traditional systems may not be able to analyze the data that is in motion (Katal, Wazid, & Goudar, 2013). The veracity describes the quality of the data. Most likely, there are some bad values in the data, for example, some values from malfunctional sensors.

In 2011, McKinsey's researchers (Manyika et al., 2011) proposed a new definition. Big data is a data set that is too large to store, manage, and analyze. This definition is not an objective definition. There is no description of any metrics related to big data. However, the definition includes an evolutionary perspective (from a time and cross-domain perspective) that illustrates what kind of dataset can be considered as big data.

The NIST believes that big data means that the volume, the velocity of acquisition, or the representation of data limits the ability to analyze and process data using existing methods (Cooper & Mell, 2012). It is necessary to use horizontal expansion mechanisms to improve the processing efficiency. Besides, there are many big data platforms for big data analysis. The big data framework is a distributed processing and analysis software library and algorithms for solving big data problems, such as Apache Spark, and Apache Kafka.

2.1.2 Big Data Analysis

In all digitized data of human beings, only a tiny part of numerical data has been well analyzed and mined, using methods such as regression, classification, and clustering. Large-scale Internet companies have conducted a preliminary analysis of semi-structured data, such as web page indexing and social data. It is difficult to analyze unstructured data effectively, such as voice, pictures, and video, which account for nearly 60% of the total. Therefore, the development of big data analysis technology needs to solve the problems in two directions. The first is to conduct a highly efficient and in-depth analysis of structured and semi-structured data and to mine tacit knowledge, such as web pages composed of natural language texts. The second is to analyze unstructured data, transform vast amounts of complex, multi-source speech, image, and video data into machine-recognizable, semantically explicit information, and then extract useful information from it.

There are two technical routes for data analysis. One is to establish mathematical models by using prior knowledge to analyze data, and the other is to build artificial intelligence systems and use a large amount of sample data for training so that machines can obtain the ability to extract knowledge from data instead of humans. Unstructured data, which is the main part of big data, is often difficult to be processed using traditional approaches. Researchers need to mine the hidden knowledge hidden in the data. It is almost impossible to build a mathematical model for analysis by hand. The analysis of big data through artificial intelligence and machine learning technology has been considered by the industry to have a good prospect. According to the hierarchical nature of the human

brain's cognitive process, researchers proposed to increase the layer of artificial neural networks and the number of neuron nodes, increase the scale of machine learning, and build a deep neural network that can improve the training effect (Dahl, Yu, Deng, & Acero, 2011). Their proposal was confirmed by the follow-up experiments. The artificial intelligence technology, which has been quiet for many years, has once again become a hot spot in data analysis technology, which has caused great interests in the industry and academia.

2.2 Feature Selection

As one of the common dimension reduction methods, feature selection is a crucial research field in pattern recognition. It refers to selecting a subset of features from full set that optimizes an evaluation criterion. The main task of feature selection is to select features to represent the system effectively.

2.2.1 Significance of Feature Selection

In general, feature selection is a searching optimization problem. For feature sets whose size is n, the search space consists of $2^n - 1$ possible states. In the real world, when the number of features is huge, it will be impracticable to conduct an exhaustive search because the amount of calculation is too large. Therefore, selecting features from the data sets is a crucial and essential procedure in big data analysis. The purpose of feature selection is to ensure the model constructed by the selected optimal feature subsets having an approximate or even better prediction accuracy compared to the one without feature selection. Feature selection can improve the generalization ability (Abe, 2010) and computational efficiency of the model (Vafaie & De Jong, 1993). The performance of some learning algorithms became worse because of irrelevant or redundant features (Weston et al., 2001). The number of training samples required for most learning algorithms increases dramatically with increasing irrelevant features (Jain & Zongker, 1997). Therefore, selecting good features not only reduces computational complexity,

improves prediction accuracy, but also helps with finding a more accurate algorithm model. The emergence of large-scale data processing problems, such as information retrieval, and genetic analysis (Xing, Jordan, Karp, et al., 2001), makes feature selection more important in data processing and machine learning.

2.2.2 Feature Selection in Big Data

With the development of various technologies, the data scale has exploded. The data volume of the data centers has reached ZB, YB level (Matsuoka et al., 2014). Most of the data is unstructured data (Sivarajah, Kamal, Irani, & Weerakkody, 2017). Also, high dimensionality could be an important feature of these data. For instance, an email-spam filtering task can have 1013 unique features (Weinberger, Dasgupta, Attenberg, Langford, & Smola, 2009). It is common that there are over 20,000 or more potential features in biological data sets (Hua, Tembe, & Dougherty, 2009). Collecting storage is only the first step of big data application, how to process the stored data, predicting the future, and providing strong support for business decision-making and scientific research are the big challenges of big data. The high dimensionality makes the process of big data require massive memory and computational cost in training. At the same time, it reduces the generalization ability due to the issue of curse of dimensionality (Donoho et al., 2000). Dimensionality reduction is an important pre-processing step for high-dimensional data. Feature selection and feature extraction are common methods for dimensionality reduction. Feature selection is the main data dimension reduction method because of its high interpretability.

2.2.3 Categories of Feature Selection

According to Guyon and Elisseeff (2003), there are three categories in the methods of feature selection: wrapper, filter, and embedded method. The wrapper method relies on specific machine learning algorithms for feature selection. It considers the system as a black box. It trains the learner directly with the selected features and evaluates the

selected features based on the learner's performance. The wrapper method iteratively evaluates the selected feature subset to obtain the best set. The brute-force nature determines that it is compute-intensive. The wrapper method is not as efficient as the filter method regarding computational efficiency, but the size of the selected subset of optimization features is relatively small, which is very beneficial to identify the key features and to simplify the structure of decision-making machine.

The difference between the wrapper method and filter method is whether the evaluation of the subset of optimized features uses the learning algorithm used in the construction of the decision machine. In the filter method, the ideas of multiple disciplines such as statistics and information theory are used to evaluate the predictive ability of every feature according to the intrinsic characteristics of data set. Several features with better ranking constitute a feature subset (Zhou, Wang, & Dougherty, 2004). The filter method usually selects a subset of features that are better predictive based on feature evaluation criteria. One of the main problems is that it does not guarantee selecting a smaller optimized feature subset, especially when the association between features and classifiers is strong. Therefore, even if the method can find an optimized subset that satisfies the condition, the subset will be larger and will contain some obvious noise features. A significant advantage of this approach is that it can quickly eliminate a portion of irrelevant noise features.

Since filter and wrapper are two complementary methods, many scholars have proposed a combined feature selection algorithm that combines these two approaches. The first step is to use the filter method for feature pre-selection, eliminating some unrelated features, and reducing dataset dimensions. Further, the second step is to perform feature selection on pre-selected feature sets using the wrapper method.

The embedded method combines feature selection with the process of training. The learning and feature selection are combined. The embedded method takes advantage of the available data. The other advantage of embedded method is that it may find the solution faster because it does not retrain a predictor from beginning for each selected subset.

2.3 Logistic Regression

Regression analysis is a type of statistical method that predicts one or more response variables based on predictor variables. Regression analysis can also be used to evaluate the effects of explanatory variables on response variables, often to explain the effect of a linear function of a set of variables on a response variable. Linear regression models and logistic regression models are basic statistical methods. Linear regression aims to find a model between explanatory variables and outcome variables and utilize it to make predictions (Gandomi & Haider, 2015). Logistic regression, is a common tool to predict the probability of binary classification problems(Minka, 2003), because it generates probabilities in the range of $[0,1]$ (Kleinbaum, Dietz, Gail, Klein, & Klein, 2002).

The following equation can be used to express logistic regression model:

$$\hat{p} = h_\theta(\mathbf{x}) = \sigma(\theta^T \cdot \mathbf{x}) \tag{2.1}$$

The $\sigma(\cdot)$ in eq. 2.1 can be expressed as the following equation:

$$\sigma(t) = \frac{1}{1+exp(-t)} \tag{2.2}$$

The binary classification prediction can be expressed as equation 2.3:

$$\hat{y} = \begin{cases} 0 & \text{if} \hat{p} < 0.5, \\ 1 & \text{if} \hat{p} \geq 0.5. \end{cases} \tag{2.3}$$

There are multiple methods to compute logistic regressions, such as NewtonRaphson, iteratively re-weighted least square methods (IRWLS), and Fisher-scoring, etc. (Yang et al., 2018).

2.4 Tools of Big Data Analysis

This section provides a discussion about two tools of big data analysis, Apache Hadoop, and Apache Spark.

2.4.1 Apache Hadoop

Apache Hadoop is a famous software framework for distributed data processing. This framework mainly implements a programming paradigm: MapReduce. With this programming paradigm, developers can easily develop distributed applications. The Hadoop platform includes MapReduce, Hadoop Distributed File System (HDFS), and other related projects such as Hive, HBase, and more. MapReduce is a programming model that is widely used in the processing of large-scale data. With this software framework, developers can quickly write distributed applications. Today, the framework has been widely used in tasks such as log analysis and massive data sorting. The MapReduce programming model utilizes a traditional algorithm paradigm, which is called "divide and conquer". The divide-and-conquer method divides the complex problem into several similar sub-problems until the sub-question is small enough to directly yield the result, and the final result obtained by re-aggregating the intermediate data is the solution to the original problem. In Hadoop, the steps for divide and conquer are implemented in Mapper and Reducer, respectively. Figure 2.1 illustrates the workflow of MapReduce in Hadoop. Hadoop MapReduce adopts a master-slave structure. A Hadoop cluster may include three types of nodes, which are master nodes, worker nodes, and client nodes. The master is the only manager in the entire cluster. The main tasks are task scheduling and status monitoring. Workers are responsible for calculating work and task status responses. The client nodes are neither master nodes nor worker nodes. The client nodes load data and submit the jobs to the cluster. In Map phase, master nodes split the input data and

divide the original problem into multiple similar sub-problems, and then hand the split data and tasks to the worker nodes for calculation. In Reduce phase, master nodes collect the intermediate results calculated by the worker nodes and aggregate them into the final result.

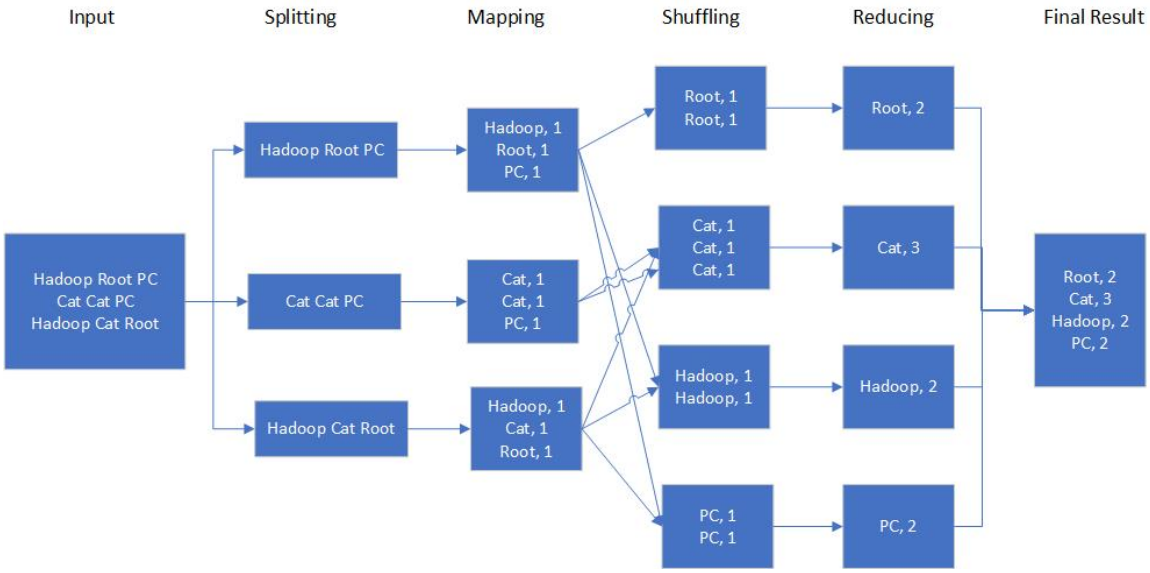

Figure 2.1. MapReduce Workflow

2.4.2 Apache Spark

Besides Hadoop, Spark is another distributed cluster computing framework for data analysis. The disadvantage of MapReduce in Apache Hadoop is that intermediate results need to be written to an external stable storage system if the computational jobs would like to reuse the intermediate results. This will incur a large number of network transmissions, disk I/O operations, and serialization, which makes the calculations inefficient. To tackle this problem, researchers have proposed different frameworks. Zaharia et al. (2012) suggested using resilient distributed datasets (RDDs) to enable in-memory computations over a large cluster. With the help of RDDs, Spark can quickly

process queries and return analysis results in real time. Compared to Hadoop, the same algorithm runs ten times to 100 times faster in Spark (Zaharia et al., 2010). Spark is technically compatible with the Hadoop storage layer APIs and can access HDFS, HBase, and SequenceFile.

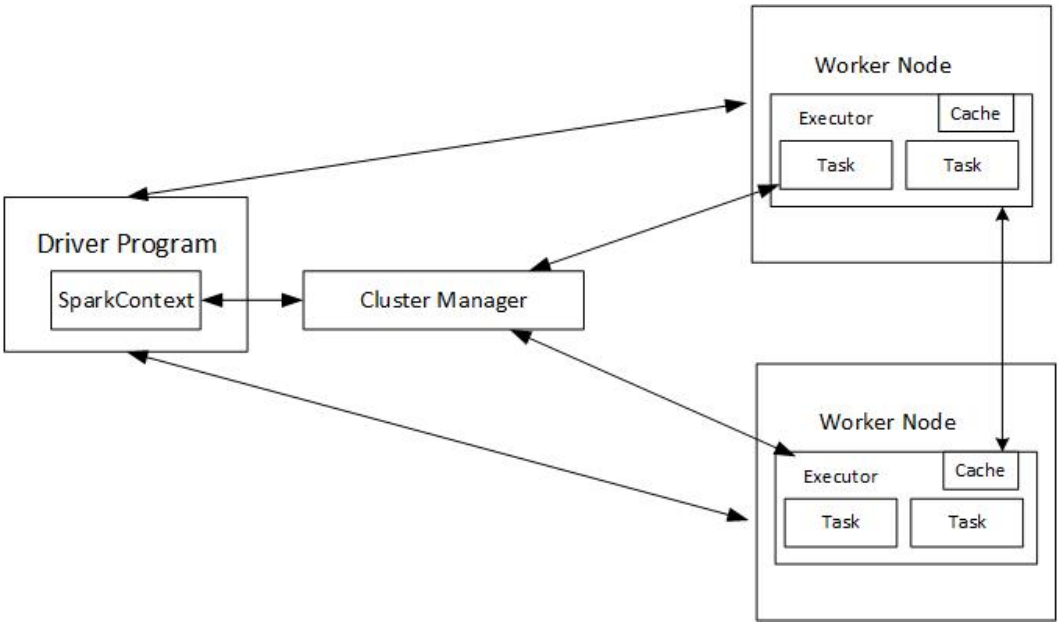

Figure 2.2. Architecture of Spark

Spark is designed for specific types of workloads in cluster computing. Specifically, Spark is suitable for the workloads that reuse working data sets between parallel operations, for example, machine learning tasks.

There are three characteristics in Spark's computing architecture.

First, Spark is a light weight cluster computing framework. Spark applies Scala to its program architecture. Scala's multi-paradigm programming language feature provides concurrency, extensibility, and support for programming paradigms.

Second, Spark includes data flow computing and interactive computing in the big data domain. Spark can interact with HDFS to get the data files. The iteration in Spark, memory computing, and interactive computing provides a good framework for data mining and machine learning.

Last but not least, Spark has a good fault tolerance mechanism. Spark uses RDDs, which are represented as a Scala object distributed over a set of nodes. In the set of read-only objects, these collections are resilient, ensuring that if part of the data is lost, it can be reconstructed.

2.5 Optimizations with the Working Sufficient Statistics

When the dataset is too large to be held in RAM, it is hard to perform data analysis. In Spark, if there is not enough room in RAM to hold the data, the data is spilled to hard drives. This approach could ensure Spark to compute dataset that is larger than the memory. However, it will compromise the performance of the computation because the data needs to be read back to the primary again in the end. To overcome these challenges, Yang et al. (2018) proposed an innovative approach to perform logistic regressions in big data. Conventionally, to select appropriate features and obtain an accurate model, multiple models need to be trained and evaluated. The data set needs to be scanned many times. Disk I/O operations are extremely expensive, especially in the big data scenarios. Therefore, reducing the disk I/O operations is crucial to big data analysis. When a dataset is much larger than the memory, a row-by-row fashion of updating the model is feasible in big data scenario.

In the study of Yang et al. (2018), they have proved that they could get an optimal prediction accuracy at the second iteration, which means that the whole data only needs to be scanned twice. The data will be read row by row. The large dataset can be represented by several small enough matrices that can be loaded into memory. Thus, even when the original dataset is larger than the memory of the computer, it is possible to perform the computation. Even when multiple feature subsets need evaluation, the disk I/O will not be the bottleneck of the process. Their result demonstrates that if the working sufficient statistics of a dataset is fetched, the computation of updating the logistic regression model is efficient.

2.6 Summary

In this chapter, a detailed literature review about big data, big data analysis, the significance of feature selection, different categories of feature selection approaches, feature selection in big data, logistic regression, Apache Hadoop, Apache Spark, optimizations with the array of working sufficient statistics is provided. The next chapter provides an introduction and discussion of the methodology of this study.

CHAPTER 3. METHODOLOGY

This chapter introduces the hypotheses and provides discussions about experiment environment, input data, working sufficient statistics and related logistic regression algorithms, testing procedures, and analysis procedures.

3.1 Hypotheses

The following are the null hypotheses and the alternative hypotheses of this study:

H_0:

- The use of working sufficient statistics algorithm does not improve the speed of feature selection and model updating in logistic regressions in Spark.
- The use of working sufficient statistics algorithm does not reduce the memory usage of feature selection and model updating in logistic regressions in Spark.

H_α:

- The use of working sufficient statistics algorithm does improve the speed of feature selection and model updating in logistic regressions in Spark.
- The use of working sufficient statistics algorithm does reduce the memory usage of feature selection and model updating in logistic regressions in Spark.

3.2 Experiment Environments

This section introduces the hardware environment and software environment of this research.

3.2.1 Hardware

Generally, as a distributed cluster-computing framework, a Spark cluster with more computers is more computationally powerful than the one with less computers. In industries, many companies have already applied Spark in production. The numbers of worker nodes in their Spark clusters are even more than 1,000. In this study, due to limited resources, the cluster used for the experiment consists of three machines: one master node, and two worker nodes. The detailed hardware specifications of three computers are listed in the following tables.

Table 3.1. Hardware Configuration of Master Machine

Component	Hardware Configuration
CPU	Intel Core i7-3770 3.4GHz
RAM	DDR3 16GB
Hard Drive	1TB 7200 rpm

Table 3.2. Hardware Configuration of Worker 1

Component	Hardware Configuration
CPU	Intel Core Q8400 2.66GHz
RAM	DDR3 4GB
Hard Drive	250GB 5400 rpm

Table 3.3. Hardware Configuration of Worker 2

Component	Hardware Configuration
CPU	Intel Core Q8400 2.66GHz
RAM	DDR3 4GB
Hard Drive	250GB 5400 rpm

3.2.2 Software

Ubuntu 16.04.6 was installed on the three machines. Apache Spark 2.3.2 was installed and set up with the default settings on the three machines. Apache Hadoop 3.1.1 was installed and set up with the default settings on the three machines. Besides, all the Spark jobs were implemented in Scala. The matrix and vector computation operations were written using breeze package. The version of Scala was 2.11.8. The version 1.8.0_191 of Java JDK was installed on the three machines. The dataset was preprocessed using Python.

Table 3.4. Software Specification

Software	Version
OS	Ubuntu 16.04.6
Spark	2.3.2
Hadoop	3.1.1
Scala	2.11.8
Java	1.8.0

3.3 Data

This study investigated a high performance computing dataset, Fresco: Open Source Data Repository for Computational Usage and Failures, a dataset of high-performance computing data collected from Purdue's Central Computing Clusters from March 2015 to June 2017 (Saurabh Bagchi, 2018). Fresco is a data repository of performance data for high-performance computing jobs that are submitted to Purdue University's Conte cluster, which consists of 580 nodes. There are around 16 tables recording performance parameters of each job's execution on an individual node. Due to the limited time, this study only investigated five tables that were possibly related to a job's failure, which were accounting, CPU, mem, process, and block. The detail descriptions of these tables are listed in Table 3.5.

Table 3.5. Fresco Dataset Information

Table	Description
Accounting	Accounting logs from job scheduler, i.e., jobID, Exit status, start time, end time, nodes and cores
Mem	Memory usage per node, i.e., total memory, memory used, FilePages, PageTables, writeback
Process	Process statistics per node, i.e., context switches, processes, 1/5/15 minute load average
CPU	CPU statistics per node, i.e., time in user mode, time in system mode, time in I/O wait
Block	Block device statistics, i.e., wait time for all requests, sectors read, wait time for read requests

There are some outliers in some features. Some features with large standard deviations were normalized. It is recommended to normalized the data before computing, because the training model is difficult to converge when the standard deviations of the data are large. Specifically, during the matrix computation, NaN errors can occur when inverting matrices with large standard deviations. The data was normalized using z-score, as following equation 3.1.

$$x_i^* = \frac{x_i - \mu}{\sigma} \tag{3.1}$$

where μ is the mean of each feature, x_i is feature value of the ith observation, and σ is the standard deviation of each feature. After normalization, the data becomes the z-score.

The dataset was downloaded to the local server. The size of the original dataset was 252GB. Before performing the experiments, the data had been preprocessed. All the entries with missing values or invalid fields had been removed from the dataset. In the CPU, Block, Mem, and Process table, most of the jobs have more than one entries. The data were collected every five minutes. The longer the job ran, the more data it generated.

The maximum and minimum were calculated to combine all the information into one row. After preprocessing, the size of the whole dataset was about 4.4 GB. The cleaned data and simulated data was saved on HDFS. There were 694 columns including onehot encoding on the 580 nodes. There were 986,374 jobs in the dataset.

The size of cleaned data of Fresco dataset was only 4.4 GB. In this study, I was also interested in examining the performance of the two methods when the input size was much larger than the memory capacity of Spark cluster. Therefore, I generated three simulated data files (10 GB, 16GB, 28GB). All the features except the Exit_status were a result of the diversion of two random numbers in the range of 5 to 15. The Exit_status is 1 or 0 with equal possiblility.

3.4 Array of Working Sufficient Statistics and Algorithms

In statistics, the following equation expresses the logistic regression model:

$$h_\theta(\mathbf{x}) = \sigma(\theta^T \cdot \mathbf{x}) \tag{3.2}$$

After initial computation, several iterations are needed to compute the logistic regression. In the iterative process, specifically, in the $i+1$ iteration, β, σ^2 , and $\mathbf{V}$ can be calculated using the following equations:

$$\beta^{(i+1)} = \{\mathbf{S}_{xx}^{(i)}\}^{-1} s_{xz}^{(i)} \tag{3.3}$$

$$\{\sigma^{\,2}\}^{(i+1)} = \frac{1}{n}\{s_{zz}^{(i)} - \{\, s_{xz}^{i}\}^T \{\mathbf{S}_{xx}^{(i)}\}^{-1} s_{xz}^{(i)}\} \tag{3.4}$$

$$\mathbf{V}^{(i+1)} = (\mathbf{X}^T \mathbf{W}^{(i)} \mathbf{X})^{-1} \tag{3.5}$$

In logistic regression, the model is only related with

$$C(\mathbf{y}, \mathbf{X}) = (c_{yy}, \mathbf{c}_{xy}, \mathbf{C}_{xx}) \tag{3.6}$$

where c_{yy} is a real number, $\mathbf{C}_{xy}$ is a vector of p-dimension, and $\mathbf{C}_{xx}$ is a matrix of dimensions $p * p$ (Yang et al., 2018). $C(\mathbf{y}, \mathbf{X})$ is the working sufficient statistics. When the p is not large, a single computer can load $C(\mathbf{y}, \mathbf{X})$ into memory and perform all the remaining computations in memory. The data was split into training data and test data. Similarly, the working sufficient statistics for testing data can be expressed as the following equation.

$$C(\mathbf{z}, \mathbf{W}) = (c_{zz}, \mathbf{c}_{wz}, \mathbf{C}_{ww}) \tag{3.7}$$

The algorithm of computing $C(\mathbf{y}, \mathbf{X})$, $\hat{\beta}$, and $\hat{\sigma^2}$ is shown in Algorithm 3.1.

Algorithm 3.1 Algorithm for Computing of $\hat{\beta}$ and $\hat{\sigma^2}$

Input: each row of input data

Output: : $\hat{\beta}$, $\hat{\sigma^2}$

function COMPUTE(*data*)

First Iteration:

Initialize $s_{zz}^{(0)}$, $\mathbf{s}_{xz}^{(0)}$, and $\mathbf{S}_{xx}^{(0)}$ with zeros

for all *row* in dataset **do**

Initialize $z_j^{(0)}$ and $w_j^{(0)}$

Compute $s_{zz}^{(0)} = s_{zz}^{(0)} + w_j^{(0)} \{z_j^{(0)}\}^2$

Compute $\mathbf{s}_{xz}^{(0)} = s_{xz}^{(0)} + w_j^{(0)} \{z_j^{(0)}\} x_j$

Compute $\mathbf{S}_{xx}^{(0)} = s_{xx}^{(0)} + w_j^{(0)} x_i x_j^T$

end for

Compute $\beta^{(1)}$, $\sigma^{2(1)}$, and $\mathbf{V}^{(1)}$ by equation 3.3, equation 3.4, and equation 3.5

Iterative Computation:

Initialize $s_{zz}^{(i)}$, $\mathbf{s}_{xz}^{(i)}$, and $\mathbf{S}_{xx}^{(i)}$ with zeros

for all *j*th row in dataset **do**

Compute $s_{zz}^{(i)} = s_{zz}^{(i)} + w_j^{(i)} \{z_j^{(i)}\}^2$

Compute $\mathbf{s}_{xz}^{(i)} = s_{xz}^{(i)} + w_j^{(i)} \{z_j^{(i)}\} x_j$

Compute $\mathbf{S}_{xx}^{(i)} = s_{xx}^{(i)} + w_j^{(i)} x_j x_j^T$

end for

Compute $\beta^{(i+1)}$, $\sigma^{2(i+1)}$, and $\mathbf{V}^{(i+1)}$ by equation 3.3, equation 3.4, and equation 3.5

Iterate step 10 to step 18 until convergence

Return the value of $\hat{\beta}$ and $\hat{\sigma^2}$

end function

There are three major procedures in algorithm 3.1. The first procedure is from line 1 to line 10. This procedure calculates the $\beta^{(1)}$. The time complexity of the first procedure is $O((p+1)^2)$. The second procedure is from line 11 to line 18. The task of this procedure calculates the $\hat{\beta}$ and $\hat{\sigma}^2$. The time complexity of the second procedure is the same as the first procedure, which is also $O((p+1)^2)$. The last procedure is to generate the final value of $\hat{\beta}$ and $\hat{\sigma}^2$, which is $O(1)$ in space complexity.

Based on the previous analysis, the space complexity of Algorithm 3.1 is $O((p+1)^2)$. In reality, p is far less than n. Therefore, there is no memory barrier for this algorithm. Even when the dataset is huge, the exact value of $\hat{\beta}$ and $\hat{\sigma}^2$ can be computed by Algorithm 3.1. As long as the data is stored in hard drives, we can read the data in a row-by-row fashion, and obtain the logistic regression model using Algorithm 3.1. This algorithm can be applied to the distributed cluster-computing systems, such as Spark, Hadoop, and other systems.

3.5 Variables

The independent variables are the logistic regression methods, including working sufficient statistics method and Spark official logistic regression method from Spark ML package, number of features, size of memory, and size of data input. The dependent variable is the performance of Spark, i.e., running time and usage of memory.

3.6 Testing Procedures

The implementation of Spark jobs was developed on a local machine and fully tested before performing the experiments. After onehot encoding on the 580 nodes, the total number of features was over 700. It is not feasible and not necessary to perform a complete search on all these features. After filtering out irrelevant features, there were 694

features in total in the dataset. To test the hypotheses, the working sufficient statistic approach and the Spark logistic regression API were compared in the experiments. In the experiments, both the working sufficient statistics model and the Spark logistic regression model were trained and evaluated on Spark. Two experiments were performed as follows.

The first experiment was implemented to investigate the efficiency of two methods about updating several models. In the experiment, five models with a different number of features were trained. The data of two methods, including running time and memory usage, was collected and analyzed. This experiment also investigated if increasing the feature size could improve the predictability of logistic regression models.

To compare the memory usage of two methods, the second experiment was performed under different ratios of memory and input size. The memory usage of the cluster was set to 2 GB for each worker node, which is 4 GB in total for the cluster. The size of the input data was in the range of 784 MB to 28 GB.

3.7 Analysis

The processing time was measured in the code of Spark job. The data of all submitted Spark jobs was recorded and stored in log files. The Spark web interface also shows the processing time, memory usage, core number, error logs, and other information of each finished job. The results were then compared and analyzed to accept or reject the null hypotheses. All the Spark jobs were repeated three times to make sure results of the experiments were reliable. The averages were calculated and used for analysis.

3.8 Summary

In this chapter, a discussion of the methodology of the study was provided. The hardware and software environments of the research were discussed in detail. The chapter also introduced the details of the dataset, testing procedures, and analysis. In the next chapter, the results of the experiments and discussion of the research were provided.

CHAPTER 4. RESULTS AND DISCUSSION

This chapter provides the results of several different experiments and the discussion of the results.

4.1 Training Models with Different Number of Features

To obtain an optimal machine learning model, it is necessary to train multiple models and choose the best model from them. In this section, five models were designed and trained using both working sufficient statistic method and Spark logistic regression method to compare the performance of the two methods.

4.1.1 Configuration of Spark Cluster

In this experiment, the configuration of the Spark cluster was set to the same. The detail specification is shown in Table 4.1.

Table 4.1. Spark Setting Parameters

Information	Value
Cluster configuration	1 master, 2 worker nodes
Total executor memory	4 GB
Driver memory	2 GB
Total executor cores	8

4.1.2 Input Data

There were two input files in this experiment. The first input file was a 4.4 GB file in the format of CSV. The other one was a 10 GB CSV file. The data was split into two parts: training data and test data. The ratio of training data and test data was 6 : 4. When the size of the input files varied, the memory setting of the Spark cluster stayed the same. The information of input data is listed in Table 4.2.

Table 4.2. Data Information

Information	Input File 1	Input File 2
Input data	fresco_onehot.csv	onehot_10G.csv
File size	4.4 GB	10 GB
Feature number	693	693
Entries	986374	986374
Training test ratio	6 : 4	6 : 4

4.1.3 Logistic Regression Models

In this experiment, five models with a different combination of features were constructed. There were 693 features in the dataset. The first model consisted of nine features, which were selected from the process table, including ctxt, process, load_1, load_5, load_15, nr_running, user, nice, and system. The second model did not include maximums and minimums. The third model consisted of only minimums and maximums from the five tables. The fourth model included all the features except the 580 onehot encoding features. The last model consists of all 693 features, which was called onehot model. The detailed information of the five models are shown in Table 4.3.

Table 4.3. Feature Information of the Five Models

Model	Features
9-feature	9 features from process
39-feature	39 features without minimums, maximums, and one-hot encoding
73-feature	73 features, including minimums and maximums
112-feature	112 features without one-hot encoding on nodes
onehot	693 features including 580 features of one-hot encoding on nodes

4.1.4 Train Multiple Models in One Program

To compare the performance difference between the two methods, the aforementioned five models were trained and tested. The results are listed in the following tables and figures.

Table 4.4. Average Training Time of Multiple Models With 4.4 GB Data

Method	9-feature	39-feature	73-feature	112-feature	onehot	Total
WSS	324.50	155.15	134.35	136.90	482.98	1233.89
Spark	114.07	119.41	132.38	132.79	173.86	672.50

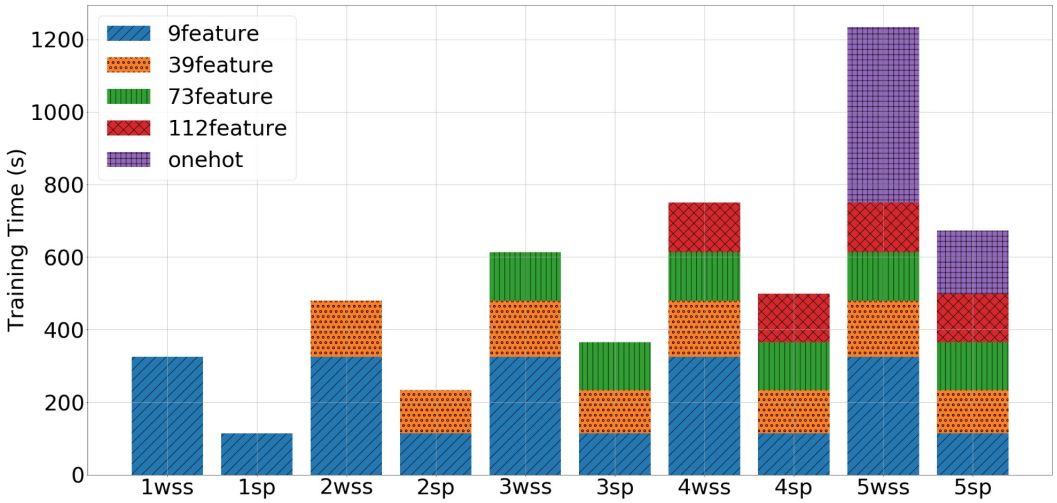

Figure 4.1. Training Time of Multiple Model with Input Size of 4.4 GB

As shown in Figure 4.1, when the number of feature in a model increased, the training time went up both for the working sufficient statistics method and the Spark API. It is expected because more features require more computations. However, the training time of the working sufficient statistics method grew faster than the Spark API method when the input data was 4.4 GB. This result indicated that the working sufficient statistics method did not outperform the Spark API when the input data was relatively small. One possible reason might be that the working sufficient statistics method involves many matrix and vector computations, such as matrix product operation and matrix inverse calculation.

Table 4.5. Average Training Time of Multiple Models With 10 GB Data

Method	9-feature	39-feature	73-feature	112-feature	onehot	Total
WSS	483. 97	144.81	138.54	126.1	542.96	1436.37
Spark	133.81	155.76	141.88	154.22	1514.69	2100.35

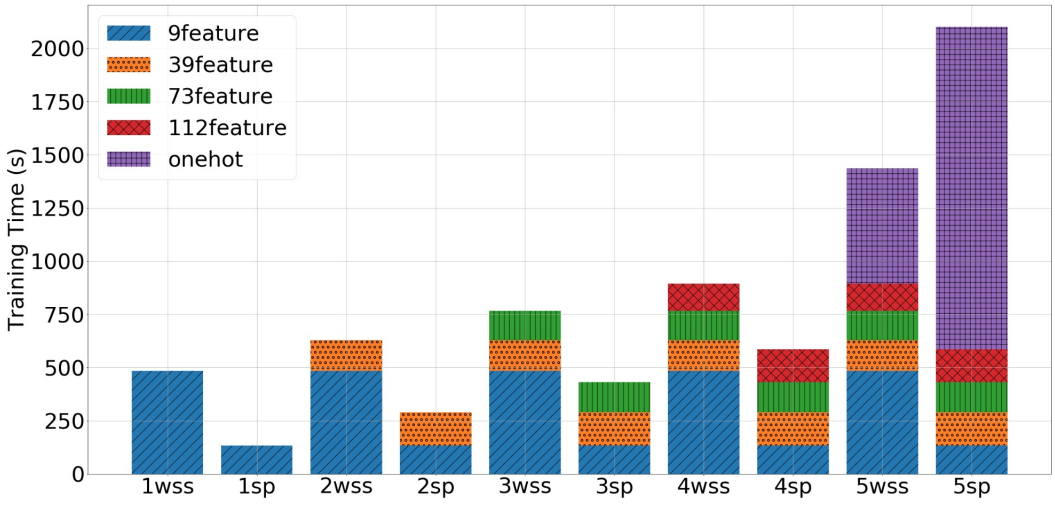

Figure 4.2. Training Time of Multiple Model with Input Size of 10 GB

As shown in Figure 4.2, in the case of 10 GB input size, when the number of features grew, both the working sufficient statistics method and the Spark API method needed more training time. However, the running time of Spark's logistic regression method increased faster than the working sufficient statistics method. The Spark's logistic regression method needed to load the full dataset into RAM. When the size of input was larger than RAM, the data was swap in/out the memory, which was time-consuming. Although the working sufficient statistics method also needed more time on models with more features, the growth of the training time was not related to the growth of the input data size.

One interesting thing is that, in the working sufficient statistics method, the training time of model with nine features was longer than the ones with 39 features, 73 features, and 112 features. For example, when the input size was 10 GB, the training time of the 9-feature model was 783.04 seconds. The training time of the 39-feature model was 213.25 seconds. This was due to the cache mechanism of Spark. The 9-feature model was the first model that was trained in our experiment. After the 9-feature model was trained, the dataset was cached in memory by Spark. The subsequent models took advantage of the cached data and needed a shorter training time than the 9-feature model.

4.1.5 Train Multiple Models in Separate Programs

When the input size is huge, it is not feasible to train multiple models in a single program. Due to the limited memory resource, in this study, one model was trained each time. In this subsection, five models were trained and tested separately. The results are shown in the following tables and figures.

Table 4.6. Average Training Time of Multiple Models in Separate Programs With 4.4 GB Data

Method	9-feature	39-feature	73-feature	112-feature	onehot	Total
WSS	100.37	138.22	134.97	152.41	492.91	1018.88
Spark	48.59	100.31	99.87	114.15	118.14	481.06

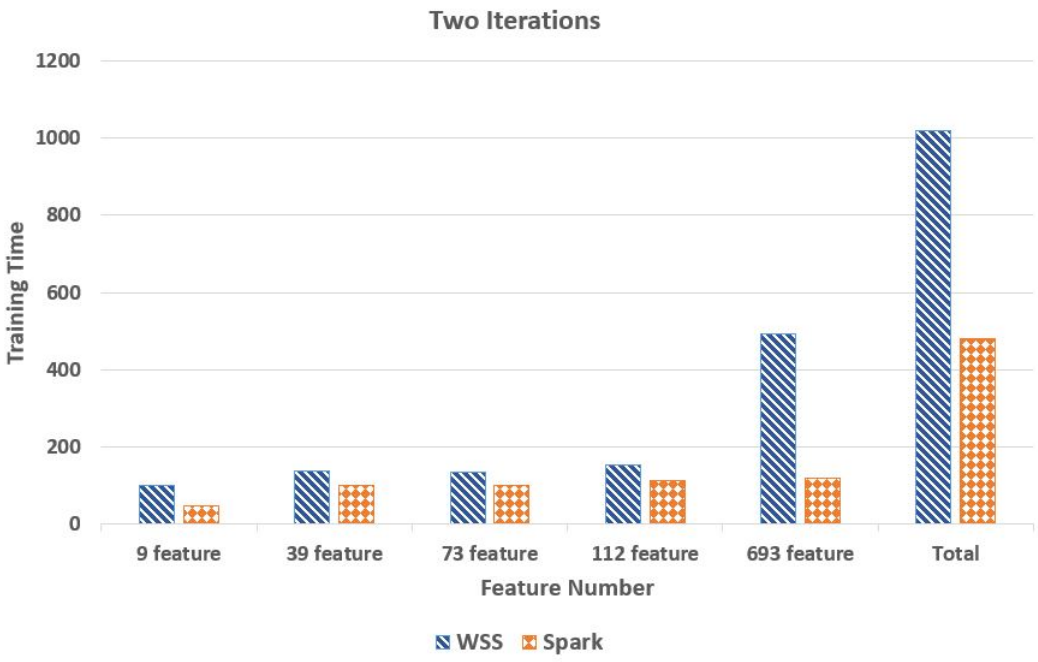

Figure 4.3. Training Time of the Five Models with Input Size of 4.4 GB

Table 4.7. Average Training Time of Multiple Models in Separate Programs With 10 GB Data

Method	9-feature	39-feature	73-feature	112-feature	onehot	Total
WSS	511.65	761.66	412.21	792.16	1381.3	3858.99
Spark	226.89	318.81	273.87	399.59	4672.29	5891.46

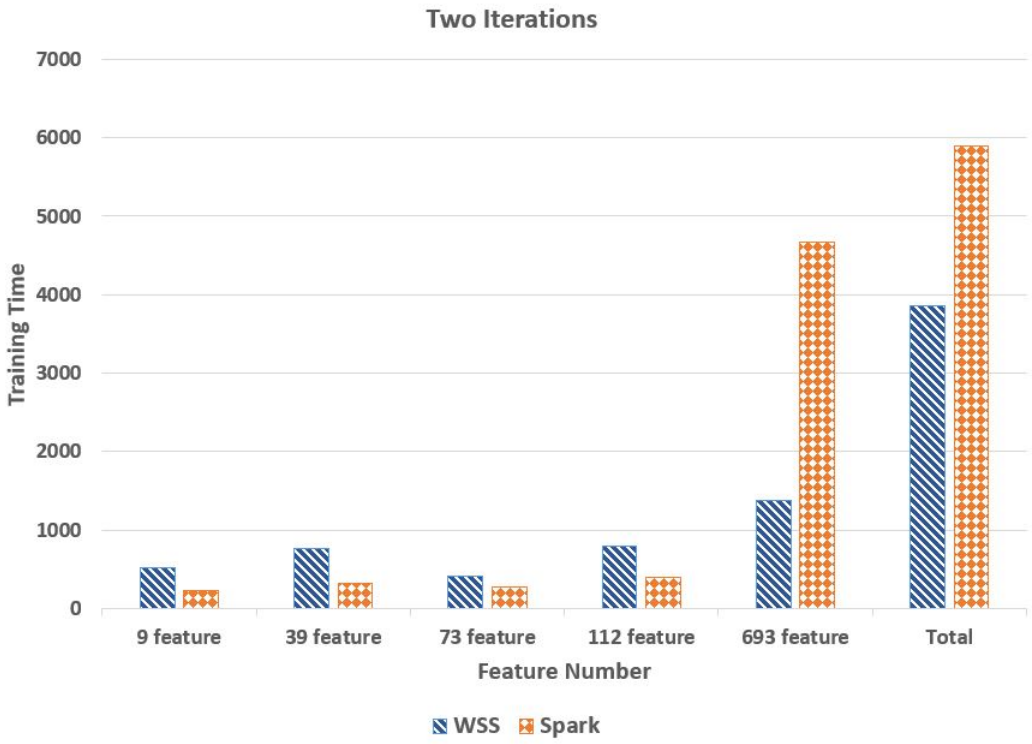

Figure 4.4. Training Time of Five Models with Input Size of 10 GB

When five models were trained separately, the training time grew when the number of features increased. When the models were trained separately, the training time of the 9-feature model was not longer than the models with more features.

4.1.6 Discussion

In this section, the experiment showed that the working sufficient statistics method could update multiple models with shorter time compared to the Spark's API when the input size was large. It is meaningful to reduce the training time because the best model is often obtained by training multiple models.

4.2 Training Logistic Regression Models With Different Ratios of Data and Memory

In this experiment, I investigated if the algorithm performed well with different sizes of the input datasets. In the two sub-experiments, the total memory of the Spark cluster was set to 2GB and 4GB, respectively.

4.2.1 Data

There were four different input files with different sizes, ranging from 784MB to 28GB. The ratio of data and memory is defined using the following equation.

$$DataMemoryRatio = \frac{InputDataSize}{SparkMemorySize} \tag{4.1}$$

The details of the four input files are listed in Table 4.8. The input files with 9 GB, 16 GB, and 28 GB were simulated data.

Table 4.8. Input Data Information

Input Size	Entries
784 MB	986374
9 GB	16186378
16 GB	27713654
28 GB	51713653

4.2.2 The Sub-experiment with 4 GB Memory

In this sub-experiment, the memory of executors in Spark was set to 4 GB in total. The details of the configuration of Spark cluster is shown in the following table.

Table 4.9. The Spark Setting Parameters for the Experiment with 4 GB Memory

Information	Value
Cluster configuration	1 master, 2 worker nodes
Total executor memory	4 GB
Driver memory	2 GB
Total executor cores	8

The ratios of data and memory are listed in Table 4.10.

Table 4.10. The Ratios of Input Data and Memory

Input Size	784 MB	9 GB	16 GB	28 GB
Ratio	0.19	2.25	4	7

The results of the experiment with 4 GB memory setting are shown in the following tables and figures.

Table 4.11. Average Training Time of Iteration 1 with 4 GB Memory

Method	784M	9G	16G	28G
WSS	43.17	329.46	1287.56	2342.53
Spark	13.56	343.09	1161.1	2218.75

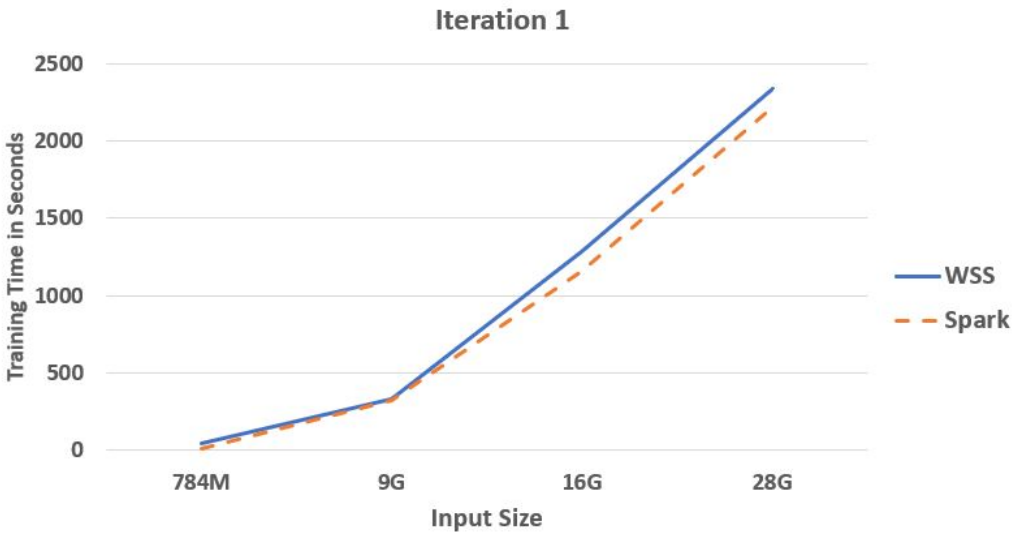

Figure 4.5. Average Training Time of Iteration 1 with 4 GB Memory

Table 4.12. Average Training Time of Iteration 2 with 4 GB Memory

Method	784M	9G	16G	28G
WSS	39.97	823.87	656.3	2753.96
Spark	18.84	976.94	779.13	2662.7

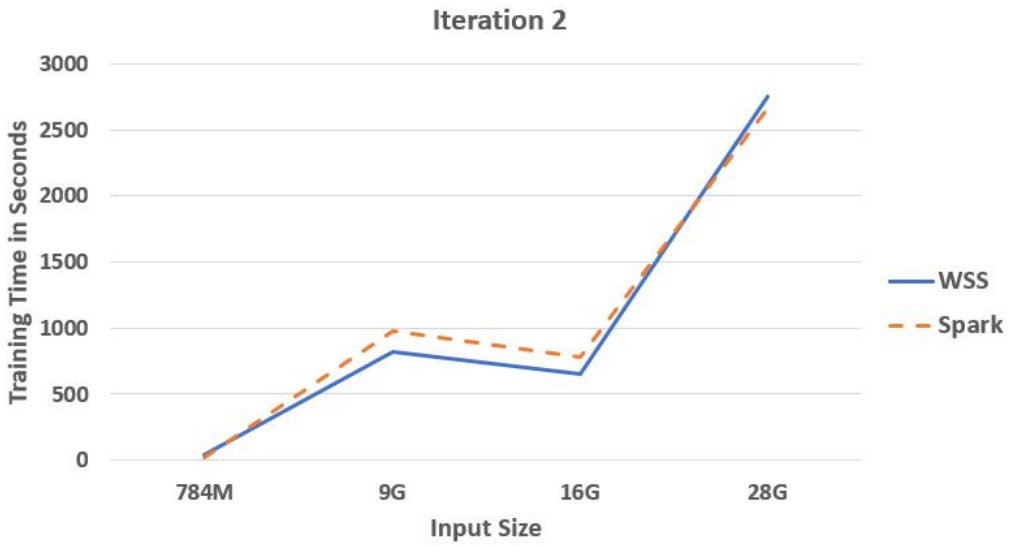

Figure 4.6. Average Training Time of Iteration 2 with 4 GB Memory

Table 4.13. Average Training Time of Iteration 3 with 4 GB Memory

Method	784M	9G	16G	28G
WSS	34.81	479.33	391.14	1742.61
Spark	17.59	999.17	886.9	3152.41

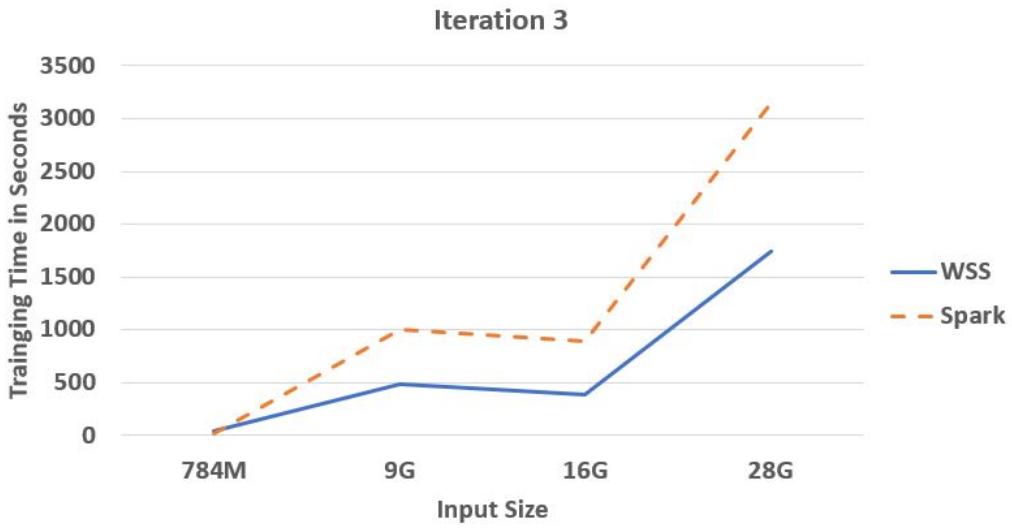

Figure 4.7. Average Training Time of Iteration 3 with 4 GB Memory

As the figures denoted, when the input data grew, both methods needed more time to train the models, especially when the input data size was 28 GB. In other words, when the ratio of data and memory was 14, the training time grew rapidly. However, the running time of Spark API grew faster than the working sufficient statistics method.

4.2.3 Sub-experiment with 2 GB Memory

To enlarge the ratio of data and memory, the total executor memory of Spark cluster was set to 2GB. The details of the configuration of Spark cluster is shown in the following table.

Table 4.14. The Spark Setting Parameters for the Experiment with 2 GB Memory

Information	Value
Cluster configuration	1 master, 2 worker nodes
Total executor memory	2 GB
Driver memory	2 GB
Total executor cores	8

The ratios of data and memory are listed in following table:

Table 4.15. The Ratios of Input Data and Memory

Input Size	784 MB	9 GB	16 GB	28 GB
Ratio	0.38	4.5	8	14

The results of the experiment with 2 GB memory setting are shown in the following tables and figures.

Table 4.16. Average Training Time of Iteration 1 with 2 GB Memory

Method	784M	9G	16G	28G
WSS	33.23	690.34	617.18	1002.38
Spark	17.8	580.25	796.79	1300.1

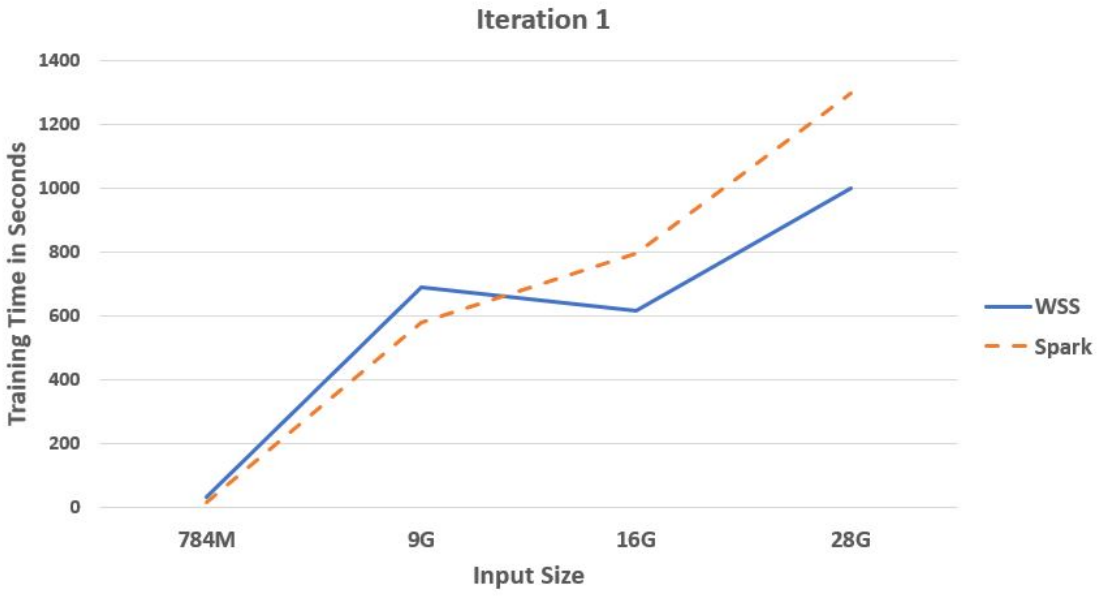

Figure 4.8. Average Training Time of Iteration 1 with 2 GB Memory

Table 4.17. Average Training Time of Iteration 2 with 2 GB Memory

Method	784M	9G	16G	28G
WSS	6.61	351.16	496.58	1059.37
Spark	13.26	742.67	814.97	1761.2

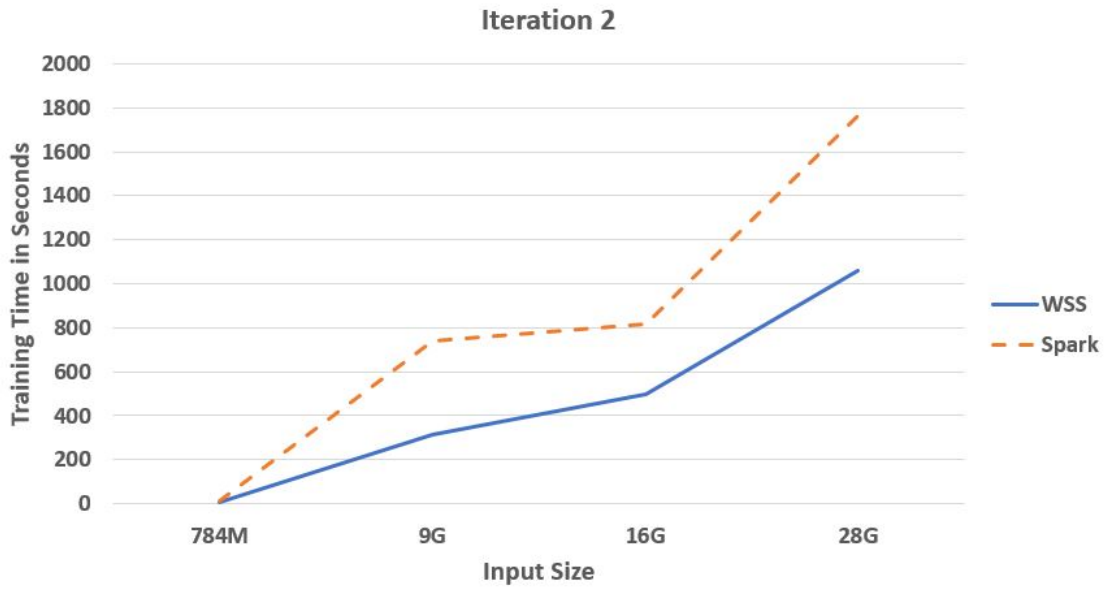

Figure 4.9. Average Training Time of Iteration 2 with 2 GB Memory

When the memory size of cluster was set to 2 GB and the input size was 28 GB, the Spark cluster could only run two iterations. During the third iteration, the Spark job was killed due to a timeout exception when running the Spark's logistic regression method. A possible reason is that the workload was overly large for the Spark cluster. When the workload was too heavy in Spark, the timeout exception could happen. When the memory size of executors was low, the Garbage Collection system might keep the Spark system very busy, which increased the workload. In this case, the ratio of data and memory was 14. When the input data size was 14 times larger than the memory size of Spark cluster, the Spark cluster might throw exceptions.

4.2.4 Discussion

In this section, the experiment reflected the advantage of the working sufficient statistics method over the Spark API. The working sufficient statistics method requires less computing memory and runs more efficiently. Although computers nowadays are equipped with large memory, it is valuable to perform classification task with less memory. After all, in the big data era, the data grows exponentially.

4.3 Model Evaluation

The section looks into the accuracy and other performance metrics of the algorithm. Accuracy, precision, recall, and F1 are frequently used metrics for model evaluation.

The evaluation scores of different models of experiment 1 are shown in the following tables.

Table 4.18. The Evaluation Scores of the 9-Feature Model

Method	Accuracy	Precision	Recall	F1
WSS	84.64%	84.64%	99.97%	91.67%
Spark API	84.61%	84.64%	99.93%	91.65%

Table 4.19. Evaluation Scores of the 39-Feature Model

Method	Accuracy	Precision	Recall	F1
WSS	84.67%	84.61%	97.94%	91.60%
Spark API	84.62%	84.62%	99.98%	91.67%

Table 4.20. Evaluation Scores of the 73-Feature Model

Method	Accuracy	Precision	Recall	F1
WSS	85.23%	84.61%	97.94%	90.79%
Spark API	84.61%	84.61%	99.97%	91.66%

Table 4.21. Evaluation Scores of the 112-Feature Model

Method	Accuracy	Precision	Recall	F1
WSS	86.47%	84.61%	96.26%	90.06%
Spark API	84.87%	84.61%	99.40%	91.41%

Table 4.22. Evaluation Scores of the Onehot Model

Method	Accuracy	Precision	Recall	F1
WSS	86.91%	84.68%	96.19%	90.07%
Spark API	86.16%	84.68%	96.21%	90.08%

As shown in the tables, the accuracy of the working sufficient statistics method was higher than the Spark API method. The recall and F1 score of the working sufficient statistics method were slightly lower than the Spark's API. Increasing the number of features in the models increased the accuracy of the models.

4.3.1 Discussion

As a logistic regression model, it is crucial that it can predict and classify correctly. According to the evaluation scores in this section, the working sufficient statistics method, has a higher accuracy score compared to the Spark existing API. The recall and F1 scores of the working sufficient statistics method were slightly lower than the Spark's API. In general, the two methods have their strengths in terms of evaluation. However, when the evaluation scores of the two methods are close, the working sufficient statistics method is able to perform the task with less memory and less time.

More interestingly, the working sufficient statistics method updates a model by reading the dataset with a row-by-row fashion. For one dataset, the obtained model and the working sufficient statistics can be saved. When there is new data, the model can be updated by just reading the new data row-by-row. This advantage makes this method suitable for processing steaming data. However, for the Spark API, updating a model with new data needs to go through the whole dataset, both the previous data and the new data.

4.4 Summary

In this chapter, the experiments were presented, including the input data, the ratios of data and memory, and the Spark cluster configuration. The results of the experiments were presented and discussed. The working sufficient statistic method and the official logistic regression method from Spark machine learning package were compared and evaluated. The working sufficient statistics method is faster than the Spark logistic regression method when updating multiple models at once. In the second experiment, the working sufficient statistics method outperforms the Spark API when the input size is much larger than the memory of Spark cluster. The next chapter presents the conclusions of the study.

CHAPTER 5. CONCLUSIONS

This thesis investigated and compared two logistic regression approaches for big data problems, which are the working sufficient statistics based logistic regression method and the Spark official logistic regression method. According to the results in Chapter 4, the two null hypotheses can be rejected. The following alternate hypotheses are accepted:

- The use of the working sufficient statistics algorithm does improve the speed of feature selection and model updating in regressions in Spark.
- The use of the working sufficient statistics algorithm does reduce the memory usage of feature selection and model updating in regressions in Spark.

Based on the results and discussion in Chapter 4, the following conclusions could be drawn:

- When the input size is large, the working sufficient statistics approach is able to perform feature selection and logistic regression tasks faster than the Spark's logistic regression API.
- The working sufficient statistics algorithm requires less memory than the Spark's existing logistic regression API. The working sufficient statistics method updates the logistic regression model by reading the data row-by-row. Therefore, its memory consumption is not affected greatly by the size of the input file. However, for the Spark's API, when the input data is greatly larger than RAM, the Spark job could be killed due to out of memory error. When the input size is equal to or smaller than the memory, the Spark's API can provide better performance than the working sufficient statistics method. When the size of input is much larger than the memory capacity of the Spark cluster, the working sufficient statistics approach can offer better performance than the existing Spark API. Although computers are equipped with larger and larger memory today and data centers consist of hundreds of computers, it is not efficient to train a model using the existing approach when the input data is extremely large.

- There is another scenario where the new approach can shine. The training results of the working sufficient statistics approach can be merged without too much effort. When the new data comes in, the Spark's logistic regression API has to retrain the model with all the data, which is time-consuming and resource-consuming. The working sufficient statistics approach can update the matrices of working sufficient statistics by only reading the new data. Therefore, this method can be utilized in the scenarios that the data grows constantly and the model needs to be updated when the new data comes in.

5.1 Future Work

In the future, the researcher will keep working on optimizing the working sufficient statistics algorithm. When the feature number of the logistic regression model is large, the training time grows rapidly. To perform the training tasks efficiently, it is necessary to further reduce the running time for the large matrices computation.

In recent years, edge computing is emerging dramatically (Satyanarayanan, 2017). In edge computing, the data is stored and processed in the edge nodes, which are close to the edge devices (Shi, Cao, Zhang, Li, & Xu, 2016). The edge nodes may be far away from each other. They could not be combined together as a cluster. The computing resource is limited in the edge nodes. Our approach may be applied in edge computing.

REFERENCES

Abe, S. (2010). Feature selection and extraction. In *Support vector machines for pattern classification* (pp. 331–341). Springer.

Angelova, A., Krizhevsky, A., & Vanhoucke, V. (2015). Pedestrian detection with a large-field-of-view deep network. In *2015 ieee international conference on robotics and automation (icra)* (pp. 704–711).

Ba, J., Mnih, V., & Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*.

Cooper, M., & Mell, P. (2012). Tackling big data. In *Federal computer security managers forum* (pp. 728–729).

Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2011). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, *20*(1), 30–42.

Donoho, D. L., et al. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, *1*(32), 375.

Dubey, R., Gunasekaran, A., Childe, S. J., Wamba, S. F., & Papadopoulos, T. (2016). The impact of big data on world-class sustainable manufacturing. *The International Journal of Advanced Manufacturing Technology*, *84*(1-4), 631–645.

Frome, A., Corrado, G. S., Shlens, J., Bengio, S., Dean, J., Mikolov, T., et al. (2013). Devise: A deep visual-semantic embedding model. In *Advances in neural information processing systems* (pp. 2121–2129).

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International journal of information management*, *35*(2), 137–144.

Gantz, J., & Reinsel, D. (2011). Extracting value from chaos. *IDC iview*, *1142*(2011), 1–12.

Ganz, F., Barnaghi, P., & Carrez, F. (2013). Information abstraction for heterogeneous real world internet data. *IEEE Sensors Journal*, *13*(10), 3793–3805.

Gupta, D., & Rani, R. (2018). A study of big data evolution and research challenges. *Journal of Information Science*, 0165551518789880.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, *3*(Mar), 1157–1182.

Hua, J., Tembe, W. D., & Dougherty, E. R. (2009). Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, *42*(3), 409–424.

Jain, A., & Zongker, D. (1997). Feature selection: Evaluation, application, and small sample performance. *IEEE transactions on pattern analysis and machine intelligence*, *19*(2), 153–158.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Katal, A., Wazid, M., & Goudar, R. (2013). Big data: issues, challenges, tools and good practices. In *2013 sixth international conference on contemporary computing (ic3)* (pp. 404–409).

Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., & Klein, M. (2002). *Logistic regression*. Springer.

Kolda, T. G., & Sun, J. (2008). Scalable tensor decompositions for multi-aspect data mining. In *2008 eighth ieee international conference on data mining* (pp. 363–372).

Lin, K. C., Zhang, K. Y., & Hung, J. C. (2014). Feature selection of support vector machine based on harmonious cat swarm optimization. In *2014 7th international conference on ubi-media computing and workshops* (pp. 205–208).

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity.

Matsuoka, S., Sato, H., Tatebe, O., Koibuchi, M., Fujiwara, I., Suzuki, S., ... others (2014). Extreme big data (ebd): Next generation big data infrastructure technologies towards yottabyte/year. *Supercomputing frontiers and innovations*, *1*(2), 89–107.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D., & Barton, D. (2012). Big data: the management revolution. *Harvard business review*, *90*(10), 60–68.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., ... others (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, *17*(1), 1235–1241.

Mills, S., Lucas, S., Irakliotis, L., Rappa, M., Carlson, T., & Perlowitz, B. (2012). Demystifying big data: a practical guide to transforming the business of government. *TechAmerica Foundation, Washington*.

Minka, T. P. (2003). A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 1–18.

Morgan, S. P., & Teachman, J. D. (1988). Logistic regression: Description, examples, and comparisons. *Journal of Marriage and Family*, *50*(4), 929–936.

Sagiroglu, S., & Sinanc, D. (2013). Big data: A review. In *2013 international conference on collaboration technologies and systems (cts)* (pp. 42–47).

Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, *50*(1), 30–39.

Saurabh Bagchi, R. K. R. K. S. H. C. E. C. S., Todd Evans. (2018). Fresco: Job failure and performance data repository from purdue university. *https://www.rcac.purdue.edu/fresco*.

Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646.

Sivarajah, U., Kamal, M. M., Irani, Z., & Weerakkody, V. (2017). Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, *70*, 263–286.

Vafaie, H., & De Jong, K. (1993). Robust feature selection algorithms. In *Proceedings of 1993 ieee conference on tools with al (tai-93)* (pp. 356–363).

Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J., & Smola, A. (2009). Feature hashing for large scale multitask learning. *arXiv preprint arXiv:0902.2206*.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2001). Feature selection for svms. In *Advances in neural information processing systems* (pp. 668–674).

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Xing, E. P., Jordan, M. I., Karp, R. M., et al. (2001). Feature selection for high-dimensional genomic microarray data. In *Icml* (Vol. 1, pp. 601–608).

Yan, X., & Su, X. (2009). *Linear regression analysis: theory and computing*. World Scientific.

Yang, B., Wang, M., Xu, Z., & Zhang, T. (2018). Streaming algorithm for big data logistic regression. In *2018 ieee international conference on big data (big data)* (pp. 2940–2950).

Yang, B., & Zhang, T. (2016). A scalable feature selection and model updating approach for big data machine learning. In *2016 ieee international conference on smart cloud (smartcloud)* (pp. 146–151).

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., ... Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th usenix conference on networked systems design and implementation* (pp. 2–2).

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. *HotCloud*, *10*(10-10), 95.

Zhao, S., Chandrashekar, M., Lee, Y., & Medhi, D. (2015). Real-time network anomaly detection system using machine learning. In *2015 11th international conference on the design of reliable communication networks (drcn)* (pp. 267–270).

Zhou, X., Wang, X., & Dougherty, E. R. (2004). Nonlinear probit gene classification using mutual information and wavelet-based feature selection. *Journal of Biological Systems*, *12*(03), 371–386.

Zwolenski, M., Weatherill, L., et al. (2014). The digital universe: Rich data and the increasing value of the internet of things. *Australian Journal of Telecommunications and the Digital Economy*, *2*(3), 47.