Journal of the Korean Data & Information Science Society
2016, **27**(4), 911–923

http://dx.doi.org/10.7465/jkdi.2016.27.4.911
한국데이터정보과학회지

RHIPE 플랫폼에서 빅데이터 로지스틱 회귀를 위한 학습 알고리즘

정병호[1] · 임동훈[2]

[12]경상대학교 정보통계학과

접수 2016년 5월 3일, 수정 2016년 5월 23일, 게재확정 2016년 7월 22일

요 약

빅데이터 시대에 머신러닝의 중요성은 더욱 부각되고 있고 로지스틱 회귀는 머신러닝에서 분류를 위한 방법으로 의료, 경제학, 마케팅 및 사회과학 전반에 걸쳐 널리 사용되고 있다. 지금까지 R과 Hadoop의 통합환경인 RHIPE 플랫폼은 설치 및 MapReduce 구현의 어려움으로 인해 거의 연구가 이루지 지지 않았다. 본 논문에서는 대용량 데이터에 대해 로지스틱 회귀 추정을 위한 두가지 알고리즘 즉, Gradient Descent 알고리즘과 Newton-Raphson 알고리즘에 대해 MapReduce로 구현하고, 실제 데이터와 모의실험 데이터를 가지고 이들 알고리즘 간의 성능을 비교하고자 한다. 알고리즘 성능 실험에서 Gradient Descent 알고리즘은 학습률에 크게 의존하고 또한 데이터에 따라 수렴하지 않는 문제를 갖고 있다. Newton-Raphson 알고리즘은 학습률이 불필요 할 뿐만 아니라 모든 실험 데이터에 대해 좋은 성능을 보였다.

주요용어: 빅데이터, 로지스틱 회귀분석, Hadoop, R, RHIPE

1. 서론

한국은 초고속 인터넷과 스마트폰의 보급률이 세계 1위이고, UN이 평가하는 전자정부 1위를 달성하는 등 공공 인프라가 우수한 나라이다. 이러한 IT 인프라를 통해 한국은 어느 나라보다 질 높은 빅데이터를 다량 보유하고 있지만, 이를 제대로 활용하지 못하는 나라로 분류되고 있다 (Davenport, 2015).

빅데이터의 등장으로 데이터마이닝 (data mining)의 중요성이 더욱 부각되고 있고 최근 들어 빅데이터 분야와 관련하여 가장 주목을 받는 분야가 바로 머신러닝 (machine learning)이다. 머신러닝은 인공지능 (artificial intelligence)의 한 분야로 컴퓨터를 인간처럼 학습시켜, 스스로 규칙을 생성하도록 하는 기술이다.

빅데이터에서 로지스틱 회귀 (logistic regression)는 머신러닝 알고리즘 중 하나로 의료, 통신, 경제학, 마케팅 등 다양한 분야에서 분류 및 예측을 위한 모형으로 폭넓게 사용되고 있다 (Hilbe, 2009).

빅데이터 분석에서 기존의 머신러닝 알고리즘을 사용하여 유용한 정보를 추출하는 것은 거의 불가능하다. 대부분의 빅데이터 머신러닝 알고리즘은 전통적인 플랫폼 상에서 실행자체가 불가능할 뿐만 아니라 하나의 컴퓨터에서 단일 스레드 (single-thread) 형식으로 실행되어 처리속도가 늦다. 따라서 빅데이터에 적합한 모형을 설계하고 구현하기 위해서는 분산처리 기반 플랫폼이 요구된다.

[1] (660-701) 경남 진주시 가좌동 900번지, 경상대학교 정보통계학과, 박사수료.
[2] 교신저자: (660-701)경남 진주시 가좌동 900번지, 경상대학교 정보통계학과, 교수 및 RINS.
E-mail: dhlim@gnu.ac.kr

Hadoop은 대용량 데이터를 분산처리 할 수 있는 자바 기반의 오픈소스 프레임 워크로서 분산파일 시스템인 HDFS (Hadoop Distributed Files System)에 데이터를 저장하고 분산처리 시스템인 MapReduce를 이용하여 데이터를 처리 한다(White, 2012).

RHIPE은 RHadoop (Prajapati, 2013; Shin 등, 2015)과 같이 R과 Hadoop의 통합환경으로 미국의 퍼듀 대학의 Saptarhis Guha (2010, 2012)가 개발한 이후 Lin 등 (2013), Prajapati (2013), Hafen등 (2014)에 의해 주로 단순한 분야에 연구가 진행되어 왔다. 국내에서는 Ko와 Kim (2013), Jung 등 (2014)이 회귀분석에 이용하였을 뿐 설치 및 MapReduce 구현의 어려움으로 인해 거의 연구가 이루어지지 않았다.

본 논문에서는 RHIPE 플랫폼에서 로지스틱 회귀 추정을 위한 대표적인 알고리즘 즉, Gradient Descent 알고리즘과 Newton-Raphson 알고리즘 (Wu와 Coggeshall, 2012)에 대해 MapReduce로 구현하고, 실제 데이터와 모의실험 데이터를 가지고 이들 알고리즘 간의 성능을 비교하고자 한다.

본 논문은 다음과 같이 구성되어 있다. 제 2절에서는 관련연구로서 Hadoop과 RHIPE, 그리고 로지스틱회귀 추정 알고리즘에 대해 간략하게 살펴보고, 제 3절에서는 RHIPE 플랫폼에서 로지스틱 회귀 추정을 위한 알고리즘의 성능에 대해 살펴보고, 제 4절에서 결론을 맺고자 한다.

2. 관련 연구

2.1. Hadoop 개요

Hadoop은 대용량 데이터를 처리할 수 있는 소프트웨어로 HDFS와 MapReduce로 구성되어 있다. HDFS는 물리적으로 네임 노드 (name node)와 데이터 노드 (data node)로 구성되어 있다. 네임노드는 파일의 디렉토리 구조, 권한과 같은 메타 정보를 저장하고, 실제 데이터는 여러 대의 데이터 노드에 분산되어 저장한다. MapReduce는 잡 트랙커 (job tracker)와 테스크 트랙커 (task tracker)로 구성되고 HDFS에 저장된 파일에서 데이터를 읽어서 가공하는 역할을 수행한다 (Jung 등, 2014; Sammer, 2012).

MapReduce의 병렬처리는 함수의 입력과 출력이 모두 <키 (key), 값 (value)> 쌍으로 구성되어 있다. Figure 2.1는 단어 세기 (word count) 예를 가지고 MapReduce의 동작 과정을 보여주고 있다. Figure 2.1에서 보면 입력데이터는 블럭 단위로 분할 (Split)되고 각 블럭은 <키, 값> 쌍으로 표현된다. 분할된 각 블럭에 대해 Map 함수 수행 후 다음 단계의 작업을 위해 <키, 값> 쌍으로 임시 저장된다. Shuffle / Sort 단계에서는 Map 함수의 결과를 키에 따라 섞고 다시 정렬한다. 각 키에 따라 값 들에 대해 병렬로 Reduce 함수를 적용한 후 <키, 값> 쌍으로 출력한다.

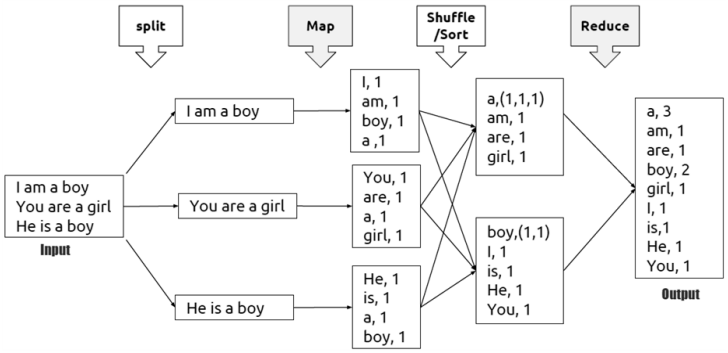

Figure 2.1 MapReduce parallel processing

2.2. RHIPE 개요

RHIPE는 R과 Hadoop이 연동하여 R 상에서 Map/Reduce 기법을 적용하여 대용량의 데이터를 통계적으로 분석할 수 있게 해주는 R 패키지이다.

RHIPE는 데이터 직렬화 (data serialization)를 위해 구글에서 제공하는 프로토콜 버퍼 (protocol buffer)를 이용한다. 프로토콜 버퍼는 언어 및 플랫폼에 대해 중립적이고 구조적 데이터로의 확장을 가능하게 해준다. 이로써 R에서 제공하는 데이터 타입을 Java, C, Python 등의 다른 개발 언어에서 사용이 가능하게 하고 다른 언어로 저장된 데이터 형태를 R로 읽어 들이는 역할을 수행한다.

Table 2.1는 RHIPE에서 제공하는 HDFS와 MapReduce 관련 함수들이다.

Table 2.1 RHIPE functions for HDFS and MapReduce operations

Function	Description
rinit()	Initialize RHIPE
rhcollect(key,value)	Generate <key, value> pairs
rhwatch()	Specify map and reduce functions and input and output directories
rhcp(ifile, ofile)	Copy files on the HDFS
rhdel(file)	Delete files on HDFS
rhls(path)	List files on HDFS
rhread(files)	Read data from HDFS into R
rhkill(job)	Stop a MapReduce job

다음은 단어 세기 예를 가지고 RHIPE에서 MapReduce 프로그램을 Map과 Reduce로 나누어 설명하고자 한다.

Map에서는 HDFS상의 데이터를 가공하여 새로운 <키, 값> 쌍의 집합을 출력하는 역할을 수행한다. 다음은 단어 세기에 대한 RHIPE에서 Map 스크립트를 나타내고 있다.

$$
\begin{aligned}
map \quad = \quad & expression(\{ \\
& words.list \leftarrow strsplit(map.values, split = \text{""}) \\
& words \leftarrow unlist(words.list) \\
& rhcollect(words, 1) \\
\})&
\end{aligned}
$$

위 스크립트에서 map 함수는 HDFS에서 불러온 데이터를 리스트 형태로 읽어 들여 처리한다. 그리고 rhcollect 함수를 이용하여 <키, 값> 쌍의 집합을 생성한 후 HDFS로 전송한다.

다음은 단어 세기에 대한 RHIPE에서 Reduce 스크립트를 나타내고 있다.

$$
\begin{aligned}
reduce \quad = \quad & expression(\\
& pre \leftarrow \{total = 0\}, \\
& reduce \leftarrow \{total \leftarrow sum(total, unlist(reduce.values)\}, \\
& post \leftarrow \{rhcollect(reduce.key, total)\} \\
)&
\end{aligned}
$$

위 스크립트에서 reduce 함수는 3개의 인자를 가지고 있다. pre 인자는 변수를 초기화하고 reduce 함수가 수행할 때 마다 실행된다. reduce 인자는 실제로 실행될 계산 작업을 정의하며 post 인자는 reduce 인자에서 계산되어진 최종 결과 값을 rhcollect 함수를 통해 HDFS로 전송하는 역할을 한다.

MapReduce 실행은 다음의 rhwatch 함수를 이용하여 map 함수와 reduce 함수를 실행시킨다.

$$rhwatch(input, input.format, map, reduce, output, combine)$$

위 rhwatch 함수는 input 인자, map 인자, reduce 인자 등 다양한 인자를 지정한다.

2.3. 로지스틱 회귀 추정 알고리즘 개요

로지스틱 회귀모형에서 독립변수 $\boldsymbol{X} = (X_0, X_1, X_2, \cdots, X_p)^T$에 대해 종속변수 Y가 $\{-1, +1\}$ 중 $Y = +1$일 확률은 다음과 같이 표현할 수 있다.

$$P(Y = +1|\boldsymbol{X}) = g(\boldsymbol{X}^T\boldsymbol{\beta}), \tag{2.1}$$

여기서 $X_0 = 1$, $\boldsymbol{\beta} = (\beta_0, \beta_1, \cdots, \beta_p)^T$이고 $g(Z) = 1/(1 + exp(-Z))$이다.

주어진 데이터 셋 $(Y_i, \boldsymbol{X}_i)$, $i = 1, \cdots, n$,에 대해 최대 우도 추정법 (maximum likelihood estimation)을 사용하여 회귀계수 $\boldsymbol{\beta}$의 추정을 위한 우도함수 (likelihood function)는 다음과 같다.

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} P(Y_i|\boldsymbol{X}_i) \tag{2.2}$$

따라서 로그-우도함수 (log-likelihood function)는 식 (2.2)에 로그를 취하고 식 (2.1)를 사용하여 다음과 같이 나타낼 수 있다.

$$\begin{aligned} \ell(\boldsymbol{\beta}) &= logL(\boldsymbol{\beta}) \\ &= \sum_{i=1}^{n} log\ g(Y_i Z_i) \end{aligned} \tag{2.3}$$

여기서 $Z_i = \boldsymbol{X}_i{}^T\boldsymbol{\beta}$이다.

우리는 Gradient Descent 알고리즘과 Newton-Raphson 알고리즘을 사용하여 식 (2.3)의 $\ell(\boldsymbol{\beta})$을 최대화하는 $\boldsymbol{\beta}$의 추정량을 구하고자 한다.

(1) Gradient Descent 알고리즘

Gradient Descent 알고리즘은 함수의 gradient에 비례하여 반복적인 방법으로 해를 찾는 방법으로 먼저 β_k에 대한 로그-우도 함수의 gradient는 다음과 같다.

$$\begin{aligned} \frac{\partial \ell(\boldsymbol{\beta})}{\partial \beta_k} &= \frac{\partial}{\partial \beta_k} logL(\boldsymbol{\beta}) \\ &= \sum_{i=1}^{n} Y_i X_{ik} g(-Y_i Z_i) \end{aligned} \tag{2.4}$$

따라서 최적해는 다음과 같이 반복적으로 업데이트함으로서 해를 찾는다.

$$\beta_k := \beta_k + \alpha \sum_{i=1}^{n} Y_i X_{ik} g(-Y_i Z_i) \tag{2.5}$$

여기서 α는 학습률을 나타낸다. α값이 너무 작으면 해에 대한 수렴속도가 늦고 α값이 너무 크면 최적해를 찾지 못하는 경우가 발생한다.

식 (2.5)에서 회귀계수를 반복적으로 업데이트함으로서 우도 (likelihood)는 증가한다. 따라서 우도는 convex 함수이므로 매번 반복에서 회귀계수의 변화가 거의 없는 경우 최대우도 값을 갖는다.

Gradient Descent 알고리즘은 간단하고 모형에 쉽게 적용이 가능한 반면에, 수렴속도가 α에 민감하게 반응하고 또한, 데이터에 따라서는 우도함수가 변수들의 범위에 크게 의존하여 수렴속도에 영향을 미친다 (Forte, 2015).

(2) Newton-Raphson 알고리즘

Newton-Raphson 알고리즘은 α가 필요없는 방법으로, 식 (2.4)에 주어진 β_k에 대한 로그-우도함수의 1차 gradient에 대해 β_j에 대한 로그-우도 함수의 2차 gradient는 다음과 같다.

$$\begin{aligned} \frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial\beta_j \partial\beta_k} &= \frac{\partial}{\partial\beta_j} \sum_{i=1}^{n} Y_i X_{ik} g(-Y_i Z_i) \\ &= -\sum_{i=1}^{n} X_{ij} X_{ik} g(Y_i Z_i) g(-Y_i Z_i) \end{aligned} \tag{2.6}$$

H를 Hessian 행렬, 즉, 로그-우도 함수의 2차 gradient 행렬이라 할 때 식 (2.6)은 H의 (j,k)번째 원소 즉, H_{jk}를 나타낸다. 따라서 최적해는 H를 사용하여 다음과 같이 반복적인 방법으로 얻어진다.

$$\beta_k := \beta_k - H^{-1} \sum_{i=1}^{n} Y_i X_{ik} g(-Y_i Z_i)$$

Newton-Raphson 알고리즘은 수렴속도는 빠르지만 매 반복마다 Hessian 행렬의 역행렬이 존재해야 되고 차원이 큰 데이터인 경우 많은 저장공간과 과중한 역행렬 계산이 요구된다.

3. RHIPE 플랫폼에서 알고리즘 성능 실험 및 분석

3.1. 실험환경

본 논문에서 사용된 실험환경은 6대의 개인용 PC를 사용하여 Figure 3.1과 같은 클러스터를 구축하였다. 6대 중 1대 PC를 마스터 (master) 노드, 나머지 5대 PC를 슬레이브 (slaves) 노드로 설정하였으며 구축된 시스템의 하드웨어 및 소프트웨어 사양은 Table 3.1과 같다.

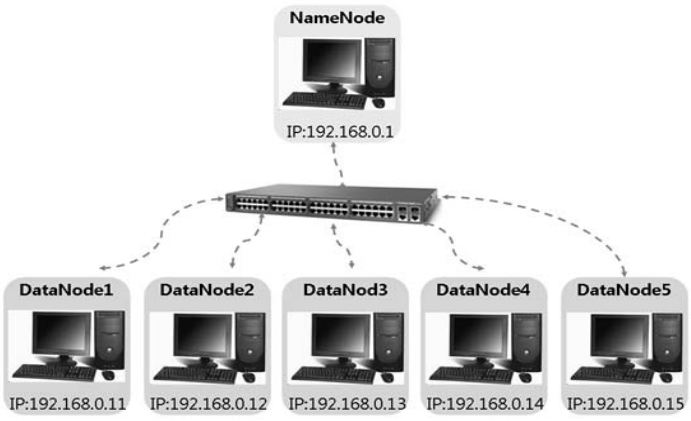

Figure 3.1 Cluster configuration with 6 nodes

Table 3.1 Experimental setup

Hardware specs	CPU		Intel(R)Core(TM)2 Duo CPU E8300@2.83GHz
	RAM	master	4 GB
		slaves	2 GB
	Network Interface Card		RealTek
	Switch Hub		Cisco Catalyst 2960 (1G Ethernet)
Software version	OS		12.04 LTS
	Java		1.7.0
	Hadoop		0.20.2
	R		3.1.0
	RHIPE		0.73.1
	Google Protocal Buffer		2.4.1

본 논문에서 시스템 처리시간은 R의 system.time() 함수에서 제공하는 elapsed time을 가지고 측정하였다. 여기서 elapsed time은 프로세스의 전체 경과시간을 나타내는데 흔히, 벽시계 시간 (wall clock time)을 의미한다. 동일한 시스템 조건하에서 패키지들의 성능을 비교하기 위해 각 프로세스 실행 시 메모리를 초기화 (clear)한 후 측정하였으며 측정오차를 최소화하기 위해 CPU의 유휴상태 (idle state)에서 시간 측정을 실시하였다 (Ciliendo 등, 2007).

3.2. 실험데이터

(1) 실제 데이터

실제 데이터는 2009년 ASA (American Standards Association: 미국 규격 협회)에서 공개된 미국 항공기 운항과 관련된 데이터이다 (ASA data expo, 2009). 이 항공기 데이터는 1987년부터 2008년까지 해마다 29개 변수에 대해 조사된 데이터이고 전체 데이터의 행은 123,534,970개이고 데이터의 크기는 12 GB정도이다. Wang 등 (2015)과 같이 로지스틱 회귀분석을 위해 원래 데이터에서 5개의 변수만 고려하였으며 종속변수인 도착지연시간 (ArrDelay)은 다음과 같이 얻었다.

$$ArrDelay = \begin{cases} 1, & \text{if } ArrDelay > 15 \\ 0, & otherwise \end{cases}$$

4개의 독립변수 중 출발시간 (DepHour)은 원래 데이터의 출발시간 (DepTime)에서 시 (Hour)부분만 사용하였고, 비행거리 (Distance)는 원래 데이터를 그대로 사용했다. 나머지 두 변수 밤 (Night)과 주말 (Weekend)은 아래와 같이 얻었다.

$$Night = \begin{cases} 1, & \text{if } 0 \le DepHour \le 5 \text{or } 20 \le DepHour \le 24 \\ 0, & otherwise \end{cases}$$

$$Weekend = \begin{cases} 1, & \text{if } 6 \le DayOfWeek \le 7 \\ 0, & otherwise \end{cases}$$

여기서 $DayOfWeek$은 1 (월요일) $\sim$ 7 (일요일) 값을 갖는다.

Table 3.2은 사용된 데이터 일부를 나타낸 것이다.

Table 3.2 A part of real airline data

DepHour (X_1)	Distance (X_2)	Night (X_3)	Weekend (X_4)	ArrDelay (Y)
20	810	1	0	0
7	810	0	0	0
6	515	0	0	0
⋮	⋮	⋮	⋮	⋮
10	1591	0	0	0
6	451	0	0	0
16	451	0	0	0

(2) 모의실험 데이터

모의실험에서 $p = 4$인 독립변수 $X_1, \cdots, X_4$는 다음과 같이 얻어진다.

$$X_j \sim N(0,1), j = 1, \cdots, 4$$

그리고 모형에서 $\beta_0, \beta_1, \cdots, \beta_4$는 다음과 같다.

$$\beta_0 = 0.7,\ \beta_1 = 1.0,\ \beta_2 = 1.3,\ \beta_3 = 0.25,\ \beta_4 = 0.05$$

여기서 β_j 들은 Rashid(2008)에서 사용한 값들이다. 따라서 로지스틱 회귀 모형은 다음과 같다.

$$g(Z) = \frac{1}{1+e^{-Z}},$$

여기서

$$Z = 0.7 + 1.0X_1 + 1.3X_2 + 0.25X_3 + 0.05X_4,$$

$$Y \sim B(1, g(Z)),$$

즉, Y는 이항분포 $B(1, g(Z))$로부터 0과 1인 값을 취한다.

3.3. 로지스틱 회귀추정 알고리즘의 MapReduce 구현

본 논문에서 논의할 로지스틱 회귀 추정 알고리즘의 MapReduce 구현에 앞서 Gradient Descent 알고리즘에 사용하는 학습률 α 결정에 대해 먼저 논의하고자 한다. 여기서는 대용량의 데이터를 고려하여 그리드 탐색법 (grid search)과 이진 탐색법 (binary search) (Tzafestas, 1992)의 혼합방식을 이용하여 결정한다. 이에 대해 단계별로 설명하면 다음과 같다.

STEP 1: $\alpha \in L = [10^{-10}, \cdots, 10^{-(i+1)}, 10^{-i}, 10^{-(i-1)}, ..., 10^{1}]$라 하자

STEP 2: 그리드 탐색법에 의해 범위 L에서 큰 수부터 차례로 수렴하는 값을 찾는다. 10^{-i}를 최초 수렴하는 값이라 할 때, $\alpha \in L_i = [10^{-i}, 10^{-(i-1)}]$이다.

STEP 3: 이진 탐색법에 의해 범위 L_i의 중간점 $\frac{10^{-i}+10^{-(i-1)}}{2}$에서 수렴여부를 조사한다.

STEP 3.1: 수렴할 경우, $\alpha \in L_i = [\frac{10^{-i}+10^{-(i-1)}}{2}, 10^{-(i-1)}]$이다.

STEP 3.2: 발산할 경우, $\alpha \in L_i = [10^{-i}, \frac{10^{-i}+10^{-(i-1)}}{2}]$이다.

STEP 4: STEP 3의 이진탐색 과정을 k번 반복하여 최종적으로 α를 결정한다. 여기서 k는 수렴성을 고려하여 $k = 4$이다.

다음으로 로지스틱 회귀 추정을 위한 Gradient Descent 알고리즘을 MapReduce로 구현하기 위한 의사코드 (pseudo-code)는 Figure 3.2과 같다. Figure 3.2에서 보는 바와 같이, 알고리즘은 크게 3 부분으로 나눈다. Map 정의 부분, Reduce 정의 부분, 그리고 반복 실행을 통해 수렴여부를 조사하는 부분이

다. 먼저 Map의 내부 구조를 보면, 입력된 전체 데이터를 여러 개의 Map으로 쪼개어 gradient를 계산한 후 Reduce로 전송되고 Reduce에서는 각 Map에서 연산된 결과 값을 합산한다. 이러한 반복과정을 통해 로지스틱 회귀계수를 추정한다.

Newton-Raphson 알고리즘에 대한 MapReduce의 의사코드는 Figure 3.3와 같다. Figure 3.3에서 보듯이, 각 Map에서 Hessian 행렬과 gradient를 계산 한 후 그 결과를 하나로 묶어 Reduce로 전송한다. Reduce에서는 각 Map의 결과를 합산한다. 이러한 반복과정을 통해 추정된 회귀계수를 얻는다.

```
1.  map phase
2.  {
3.      M ← {y_j, x_j1, x_j2, ..., x_jk}
4.      X ← {x_j1, x_j2, ..., x_jk}
5.      Y ← {y_j}
6.      delta ← Y_j • X_jk • (1/(1+exp(− Y_j • (β_k^t × X_jk))))
7.      outputlist ≪ (NULL, delta)
8.  }
9.  reduce phase
10. {
11.     outputlist ← outputlist from mappers
12.     for all delta ∈ outputlist
13.         sumofdelta += delta
14.     end for
15.     outputlist ≪ (NULL, sumofdelta)
16. }
17. Repeat until convergence(ε = 10^−3)
18. {
19.     delta ← excute MapReduce
20.     β^(t+1) = β^(t) + α • delta
21. }
```

Figure 3.2 Psudo-code for Gradient Descent algorithm

```
REQUIRE
  · Input (key, value) where key = null, and value = objects
  · A set of objects of {y_j, x_j1, x_j2, ..., x_jk} in each mapper
  · initial set of weights β = {β_0, β_1, β_2, ... ,β_k}

1.  map phase
2.  {
3.      M ← {y_j, x_j1, x_j2, ..., x_jk}
4.      X ← {x_j1, x_j2, ..., x_jk}
5.      Y ← {y_j}
6.      Hessian ←
          X_jk^t • 1/(1+exp(− Y_j • (X_jk × β_k^t))) • 1/(1+exp(Y_j • (X_jk × β_k^t))) × X_jk
7.      delta ← Y_j • X_jk^t • 1/(1+exp(Y_j • (X_jk × β_k^t)))
8.      outputlist ≪ (NULL, (Hessian , delta))
9.  }
10. reduce phase
11.     outputlist ← outputlist from mappers
12.     for all HessianDelta ∈ outputlist
13.         sumofHessianDelta += HessianDelta
14.     end for
15.     outputlist ≪ (NULL, sumofHessianDelta)
16. }
17. repeat until convergence(ε = 10^−3)
18. {
19.     resultofMR ← excute MapReduce
        Hessian , delta ← split resultofMR
20.     β^(t+1) = β^(t) − Hessian^−1 × delta
21. }
```

Figure 3.3 Psudo-code for Newton-Raphson algorithm

3.4. 로지스틱 회귀추정 알고리즘 성능비교

로지스틱 회귀 추정을 위한 알고리즘의 성능은 회귀계수 추정하는데 걸린 시간과 추정된 회귀계수의 정확도를 가지고 비교하고자 한다.

(1) 실제 데이터

3.2절의 실제 데이터에 Gradient Descent 알고리즘 적용결과, 일정시간 내에 회귀계수가 수렴하지 못하였다. 이것은 실제 데이터 변수들의 범위가 너무 다양하여 Gradient Descent 알고리즘에서 사용된 우도함수가 이들 변수에 의해 영향을 받기 때문이다 (Jung, 2016; Forte, 2015). 따라서 우리는 실제 데이터를 변수 스케일링 (Arnulf 등, 2003) 하여 스케일 된 데이터에 대해 알고리즘을 적용하였다. 여기서 변수 스케일닝은 변수들에 대한 표준화를 의미한다. 물론, 변수 스케일닝 또한 MapReduce로 구현하였다. Table 3.3은 데이터 크기에 따른 알고리즘별 계산시간을 나타낸다.

Table 3.3 Comparisons of two algorithms with scaled real data in terms of computational time in seconds

Data size	Gradient Descent		Newton-Raphson
	α	time	
50 MB	1.000e-6	529.915	409.696
100 MB	5.000e-7	613.173	460.162
200 MB	2.500e-7	623.447	480.512
300 MB	1.250e-7	909.808	481.084
500 MB	1.000e-7	714.655	561.846
1.00 GB	4.375e-8	1688.960	1048.418
1.45 GB	3.125e-8	2094.634	1465.209
2.90 GB	1.875e-8	2793.784	2408.627
5.80 GB	9.375e-9	5378.969	4648.837
11.60 GB	4.375e-9	11450.410	8817.240

Table 3.3에서 보듯이 Gradient Descent 알고리즘은 데이터 크기에 따라 α값이 서로 다름을 알 수 있다. 계산시간을 보면 Newton-Raphson 알고리즘이 Gradient Descent 알고리즘보다 빠르게 나타났다.

Table 3.4 Comparisons of two algorithms to glm function in terms of estimated logistic regression with scaled real data

Data size	Algorithms	Estimated logistic regression $g(Z) = 1/(1 + exp(-Z))$
50 MB	glm	$Z = -1.3780 + 0.0222X_1 + 0.1113X_2 + 0.1967X_3 - 0.0735X_4$
	Gradient Descent	$Z = -1.3774 + 0.0222X_1 + 0.1111X_2 + 0.1965X_3 - 0.0732X_4$
	Newton- Raphson	$Z = -1.3780 + 0.0222X_1 + 0.1113X_2 + 0.1968X_3 - 0.0735X_4$

Table 3.4은 12 GB의 실제 데이터 중 비복원 랜덤 추출 (random sampling without replacement)에 의해 뽑힌 50 MB 크기의 데이터에서 기존의 glm (generalized linear model) 함수와 비교를 위해 두 알고리즘의 추정된 회귀계수를 나타내는 표이다. Table 3.4로부터 Newton-Raphson 알고리즘은 Gradient Descent 알고리즘보다 glm 함수와 비교하여 추정된 회귀계수에서 차이가 거의 없음을 알 수 있다.

(2) 모의실험 데이터

RHIPE 플랫폼에서 모의실험 데이터에 대한 2가지 알고리즘의 실험 결과는 Table 3.5와 같다. Table 3.5에서 보듯이 실제 데이터에 대한 Table 3.3과는 다르게 Gradient Descent 알고리즘이 Newton-Raphson 알고리즘보다 빠른 처리 속도를 보이는 것처럼 보인다.

Table 3.5 Comparisons of two algorithms with simulated data in terms of computational time in seconds

Data size	Gradient Descent		Newton-Raphson
	α	time	
50 MB	1.250e-5	348.144	390.176
100 MB	6.250e-6	368.552	450.609
200 MB	3.125e-6	348.713	400.532
300 MB	1.875e-6	298.916	371.197
500 MB	1.250e-6	379.795	452.076
1.00 GB	6.250e-7	653.261	786.240
1.45 GB	3.750e-7	916.893	1120.610
2.90 GB	1.875e-7	1463.495	1687.665
5.80 GB	9.375e-8	2728.827	3247.046
11.60 GB	5.000e-8	5177.333	6221.431

Table 3.6는 모의실험 데이터에서 알고리즘들에 대한 회귀계수 추정의 정확도를 보기 위해 glm 함수와 비교한 표이다. Table 3.6에서 보듯이 Newton-Raphson 알고리즘은 glm 함수와 비교하였을 때 회귀계수의 차이가 발생하지 않았으나, Gradient Descent 알고리즘은 회귀계수에서 차이가 많이 나는 것을 알 수 있다. 그 이유는 Gradient Descent 알고리즘은 적절하지 않은 α 사용으로 정확한 회귀계수를 추정하지 못했기 때문으로 판단된다.

Table 3.6 Comparisons of two algorithms to glm function in terms of estimated logistic regression with simulated data

Data size	Algorithms	Estimated logistic regression $g(Z) = 1/(1 + exp(-Z))$
50 MB	glm	$Z = 0.7041 + 1.0045X_1 + 1.3036X_2 + 0.2491X_3 + 0.0518X_4$
	Gradient Descent	$Z = 0.7050 + 1.0057X_1 + 1.3052X_2 + 0.2494X_3 + 0.0518X_4$
	Newton-Raphson	$Z = 0.7041 + 1.0045X_1 + 1.3037X_2 + 0.2491X_3 + 0.0518X_4$

우리는 두 알고리즘간의 정확한 비교를 위해 Gradient Descent 알고리즘에서 적절한 α을 사용하여 회귀계수 추정하는데 걸린시간과 Newton-Raphson 알고리즘의 계산시간과 비교하고자 한다. 이를 위해 먼저 Gradient Descent 알고리즘에서 적절한 α값은 순차 탐색법 (sequential search)사용하여 glm 함수에 의해 추정된 회귀계수와 가장 근접한 값을 찾는다. Table 3.7은 적절한 α값을 사용하여 얻은 추정된 로지스틱 회귀식을 나타내고 있다. Table 3.7을 보면 Gradient Descent 알고리즘에서 Table 3.6보다 glm 함수의 회귀계수와 거의 일치함을 알 수 있다. Table 3.8은 Table 3.7에서 얻어진 추정된 로지스틱 회귀식을 얻는데 걸린 시간을 나타내고 있다.

Table 3.7 Comparisons of two algorithms using exact α to glm function with simulated data in terms of estimated logistic regression

Data size	Algorithms	α	Estimated logistic regression $g(Z) = 1/(1 + exp(-Z))$
50 MB	glm	-	$Z = 0.7041 + 1.0045X_1 + 1.3036X_2 + 0.2491X_3 + 0.0518X_4$
	Gradient Descent	7.7e-6	$Z = 0.7041 + 1.0046X_1 + 1.3037X_2 + 0.2491X_3 + 0.0518X_4$
	Newton-Raphson	-	$Z = 0.7041 + 1.0045X_1 + 1.3037X_2 + 0.2491X_3 + 0.0518X_4$

Table 3.8을 보면 예상하였듯이 Newton-Raphson 알고리즘이 Gradient Descent 알고리즘보다 빠른 처리 속도를 보였다.

Table 3.8 Comparisons of two algorithms using exact α in terms of computation time in seconds

Data size	Algorithms	
	Gradient Descent	Newton-Raphson
50 MB	420.244	390.176

결론적으로 말하면, 실제 데이터와 모의실험 데이터 모두에서 Newton-Raphson 알고리즘은 Gradient Descent 알고리즘보다 처리속도가 빠르다.

4. 결론

빅데이터 시대에 머신러닝의 중요성은 더욱 부각되고 있고 머신러닝에서 분류 방법의 하나인 로지스틱 회귀는 의료, 경제학, 마케팅 및 사회과학 전반에 걸쳐 널리 사용되고 있다.

오늘날 기하급수적으로 증가하는 대용량 데이터를 기존의 전통적인 플랫폼 상에서 처리 및 분석하는 것은 거의 불가능하다. 최근 R은 Hadoop의 분산처리 환경에서 빅데이터 분석도구로서 널리 사용되고 있다. R과 Hadoop의 통합환경으로 대표적 플랫폼이 RHIPE이다. 지금까지 RHIPE에 대한 연구는 설치와 MapReduce 구현하는데 어려움으로 인해 거의 이루어지지 않았다.

본 논문에서는 RHIPE 플랫폼에서 분산처리 기반의 로지스틱 회귀 추정을 위한 Gradient Descent 알고리즘과 Newton-Raphson 알고리즘을 MapReduce로 구현하고 실제 데이터와 모의실험 데이터에서 알고리즘 성능을 비교하였다. 이를 위해 Gradient Descent 알고리즘에서 학습률은 빅데이터를 고려하여 그리드 탐색법과 이진 탐색법의 혼합방식을 이용하여 결정하였고 변수 표준화를 통한 변수 스케일링 작업 또한 MapReduce 프로그램으로 구현하여 수행하였다.

두 알고리즘 성능비교 실험 결과, Gradient Descent 알고리즘은 적용이 용이하나 학습률에 민감하게 반응하고 또한, 실제 데이터에서는 수렴성에 문제점을 갖고 있었다. 본 논문에서는 변수 스케일링을 통해 수렴성 문제를 해결하였으나 기존의 glm 함수와 비교하여 추정된 로지스틱 회귀계수의 정확도와 처리속도에 대한 문제점은 여전히 갖고 있었다. 이에 반하여 Newton-Raphson 알고리즘은 사전 학습률이 필요없고 또한, 실제 데이터에서 Gradient Descent 알고리즘이 갖는 수렴성 문제도 없을 뿐만 아니라 모든 실험 데이터에 대해 좋은 성능을 보였다.

References

Arnulf, B. A., Graf, Alexander J. S. and Borer, S. (2003). Classification in a normalized feature space using support vector machines. *IEEE*, **14**, 597-605.

ASA data expo. (2009). `http://stat-computing.org/dataexpo/2009/the-data.html`.

Ciliendo, E., Kunimasa, T. and Braswell, B. (2007). *Linux performance and tuning guidelines*, IBM redbooks, IBM, International Technical Support Organization, USA.

Davenport, T. (2015). B.I.G. Forum 2015. *Big data initiative Gyeonggi*, Gyeonggi Creative Economy & Innovation Center, Gyeonggi Province, Korea.

Forte, R. M. (2015). *Mastering predictive analytics with R*, Packt Publishing Ltd, Birmingham, UK.

Guha, S. (2010). *Computing environment for the statistical analysis of large and complex data*, Ph. D. Thesis, Department of Statistics, Purdue University, West Lafayette, Indiana, USA.

Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B. and Cleveland, W. S. (2012). Large complex data: Divide and recombine (D&R) with RHIPE. *Stat*, **191**, 53-67

Hafen, R., Gibson, T., Dam, K. K. and Critchlow. T. (2014). *Power grid data analysis with R and Hadoop. in data mining applications with R*, 1-34.

Hilbe, J. M. (2009). *Logistic regression models*, Chapman & Hall/CRC Press, Florida, USA.

Jung, B. H., Shin, J. E. and Lim, D. H. (2014). Rhipe platform for big data processing and analysis, *The Korean Journal of Applied Statistics*, **27**, 1171-1185.

Jung, B. H. (2016). *A study on machine learning algorithms using distributed processing system of big data*, Ph. D. Thesis, Gyeongsang National University, Jinju, Korea.

Ko, Y. and Kim, J. (2013). Analysis of big data using Rhipe. *Journal of the Korean Data & Information science Society*, **24**, 975-987.

Lin, H., Yang, S. and Midkiff, S. P. (2013). *RABID - A general distributed R processing framework targeting large data-set problems*, IEEE International Congress on Big Data, Santa Clara, CA, USA.

Prajapati, V. (2013). *Big data analytics with R and Hadoop*, Packt Publishing Ltd, Birmingham, UK.
Rashid, M. (2008). *Inference on logistic regression*, Ph. D. Thesis, Bowling green state university, Ohio, USA.
Sammer, E. (2012). *Hadoop Operations*, O'Reilly Media, Inc., Sebastopol, CA.
Shin, J. E., Jung, B. H. and Lim, D. H.. (2015). Big data distributed processing system using RHadoop. *Journal of the Korean Data & Information science Society*, **26**, 1155-1166.
Tzafestas, A. G. (1992). *Robotic systems: Advanced techniques and applications*, Kluwer Academic Publishers, Dordrecht, Netherlands.
Wang, C., Chen, M. H., Schifano, Wu, J. and Yan, J. (2015). *A survey of statistical methods and computing for big data*, Cornell university library, Available at `http://de.arxiv.org/abs/1502.07989v1`.
White, T. (2012). *Hadoop: The definitive guide*, O'Reilly Media, Inc., Sebastopol, CA.
Wu, J. and Coggeshall, S. (2012). *Foundations of predictive analytics*, Chapman and Hall/CRC Press, Florida, USA.

Journal of the Korean Data & Information Science Society
2016, **27**(4), 911–923

http://dx.doi.org/10.7465/jkdi.2016.27.4.911
한국데이터정보과학회지

Learning algorithms for big data logistic regression on RHIPE platform

Byung Ho Jung[1] · Dong Hoon Lim[2]

[12]Department of Information and Statistics, Gyeongsang National University

Received 3 May 2016, revised 23 May 2016, accepted 22 July 2016

Abstract

Machine learning becomes increasingly important in the big data era. Logistic regression is a type of classification in machine leaning, and has been widely used in various fields, including medicine, economics, marketing, and social sciences. Rhipe that integrates R and Hadoop environment, has not been discussed by many researchers owing to the difficulty of its installation and MapReduce implementation. In this paper, we present the MapReduce implementation of Gradient Descent algorithm and Newton-Raphson algorithm for logistic regression using Rhipe. The Newton-Raphson algorithm does not require a learning rate, while Gradient Descent algorithm needs to manually pick a learning rate. We choose the learning rate by performing the mixed procedure of grid search and binary search for processing big data efficiently. In the performance study, our Newton-Raphson algorithm outpeforms Gradient Descent algorithm in all the tested data.

Keywords: Big data, Hadoop, logistic regression, R, RHIPE.

[1] Ph.D Student, Department of Information Statistics, Gyeongsang National University, Jinju 660-701, Korea.
[2] Corresponding author: Professor and RINS, Department of Information Statistics, Gyeongsang National University, Jinju 660-701, Korea. E-mail: dhlim@gnu.ac.kr