Rcpp and CUDA backend Logistic Regression

CS628 Parallism Algorithm

Yifan Duan & Eric Xin Zhou

May 6, 2015

Outline

1. **Intro**

2. Logistic Regression Algorithm

3. Result

Background

The Rcpp and CUDA provide give us the possibility that we can wrap C++ code or CUDA code in R Package. Therefore, building a parallel framework for machine learning problem will improve the performance of several algorithm of certain R Package

Existed Parallel Framework

There are several packages combine CUDA or MPI into R
Packages

- **Rmpi**

Existed Parallel Framework

There are several packages combine CUDA or MPI into R Packages

- **Rmpi**
- **R/parallel**

Existed Parallel Framework

There are several packages combine CUDA or MPI into R
Packages

- **Rmpi**
- **R/parallel**
- **Multicore**

Existed Parallel Framework

There are several packages combine CUDA or MPI into R Packages

- **Rmpi**
- **R/parallel**
- **Multicore**
- **gputools**

Existed Parallel Framework

There are several packages combine CUDA or MPI into R
Packages

- **Rmpi**
- **R/parallel**
- **Multicore**
- **gputools**
- **CudaBayesreg**

Existed Parallel Framework

There are several packages combine CUDA or MPI into R
Packages

- **Rmpi**
- **R/parallel**
- **Multicore**
- **gputools**
- **CudaBayesreg**
- **RCUDA**

Rcpp Solution

with Rcpp we can integrate the .cu Code directly into package's dynamic linking library, and the special Makefile in Rcpp Package is a Makevars

```
PKG_LIBS = $(shell $(R_HOME)/bin/Rscript -e "Rcpp::LdF
CUDA_INCS = -I/usr/local/cula/include -lcublas

GLMOBJECTS=\
./GLM/*.o \
...

RCPPGLM = RcppGLM.o

RCPP_EXPORT = RcppExports.o

OBJECTS = $(GLMOBJECTS) $(RCPPGLM) $(RCPP_EXPORT)
```

Outline

multinomial logistic regression problem

In the multinomial logistic regression problem, we have the probability definition of each class on each observation.

$$\ln(\frac{\pi_{ij}}{\pi_{iJ}}) = X\beta_j$$

multinomial logistic regression problem

- Here β_j means the j th column of matrix β.

multinomial logistic regression problem

- Here β_j means the j th column of matrix β.
- Since $\sum_{j=1}^{J} \pi_{ij} = 1$, which means that for each response Y_i, the summed probability in predictors space is 1.

multinomial logistic regression problem

- Here β_j means the j th column of matrix β.
- Since $\sum_{j=1}^{J} \pi_{ij} = 1$, which means that for each response Y_i, the summed probability in predictors space is 1.
- In Logistical regression, since we have N obvervation **Y**, and we use **Y**$_i$ represents each observation $(0, 0, 0, ..1, ..0)$.

multinomial logistic regression problem

- Here β_j means the j th column of matrix β.
- Since $\sum_{j=1}^{J} \pi_{ij} = 1$, which means that for each response Y_i, the summed probability in predictors space is 1.
- In Logistical regression, since we have N obvervation $\mathbf{Y}$, and we use $\mathbf{Y}_i$ represents each observation $(0, 0, 0, ..1, ..0)$.
- Therefore, we can define the probability of all classes:

probability of multinomial logistic regression problem

$$\pi_{ij} = \frac{e^{X_i \beta_j}}{1 + \sum_{j=1}^{J-1} e^{X_i \beta_j}} \qquad j \neq J$$

$$\pi_{iJ} = \frac{1}{1 + \sum_{j=1}^{J-1} e^{X_i \beta_j}} \qquad j = J$$

object function and gradient

$$L(\beta) = -\sum_{i=1}^{N}[\sum_{j=1}^{J-1}\mathbf{Y}_{ij}\sum_{k=1}^{p}X_{ik}\beta_{kj} - n_i[\log[1 + \sum_{j=1}^{J-1}e^{\sum_{k=1}^{p}X_{ik}\beta_{kj}}]]$$

$$\nabla = -\sum_{i=1}^{N}[\mathbf{Y}_{ij}X_{ik} - n_iX_{ik}\pi_{ij}] = -X^T(Y - \pi)$$

In one vs all model

We also can define the Hessian matrix for the update:
$\beta^{(t+1)} = \beta^{(t)} - \alpha H^{-1} \nabla.$

And in multinomial model, for each class j, we will have

$$\mathbf{H}_{kk'} = \sum_{i=1}^{N} n_i X_{ik} X_{ik'} \pi_{ij} (1 - \pi_{ij})$$

Parallel

As the result, in gradient descent, we can paralyze them in several parts:

- Summed Loss fucntion L: distribute the $[\log[1 + \sum_{j=1}^{J-1} e^{\sum_{k=1}^{p} X_{ik}\beta_{kj}}]]$ into each threads, and use **reduce to sum them.**

Parallel

As the result, in gradient descent, we can paralyze them in several parts:

- Summed Loss fucntion L: distribute the $[\log[1 + \sum_{j=1}^{J-1} e^{\sum_{k=1}^{p} X_{ik}\beta_{kj}}]]$ into each threads, and use **reduce to sum them.**
- Matrix Multiplication: in gradient function ∇, calculate the $-X^T(Y - \pi)$

Outline

1. Intro

2. Logistic Regression Algorithm

3. **Result**

Classification result

Speed up on CUDA about 8x and Classification error ~ 0.13