A study of optimization algorithms on a breast cancer diagnosis dataset

Xinlei Chen, Guojing Wu and Yujing Yao

March 14, 2019

Abstract

This report discusses a study of optimization algorithms on breast cancer diagnosis dataset. Our goal is to build a predictive model based on logistic regression to facilicate cancer diagnosis, and we compared estimation method including Newton Raphson, Gradient Decent and Lasso. Our result shows that.....

Background

The goal of the exerise is to build a predictive model based on logistic regression to facilicate cancer diagnosis.

Dataset

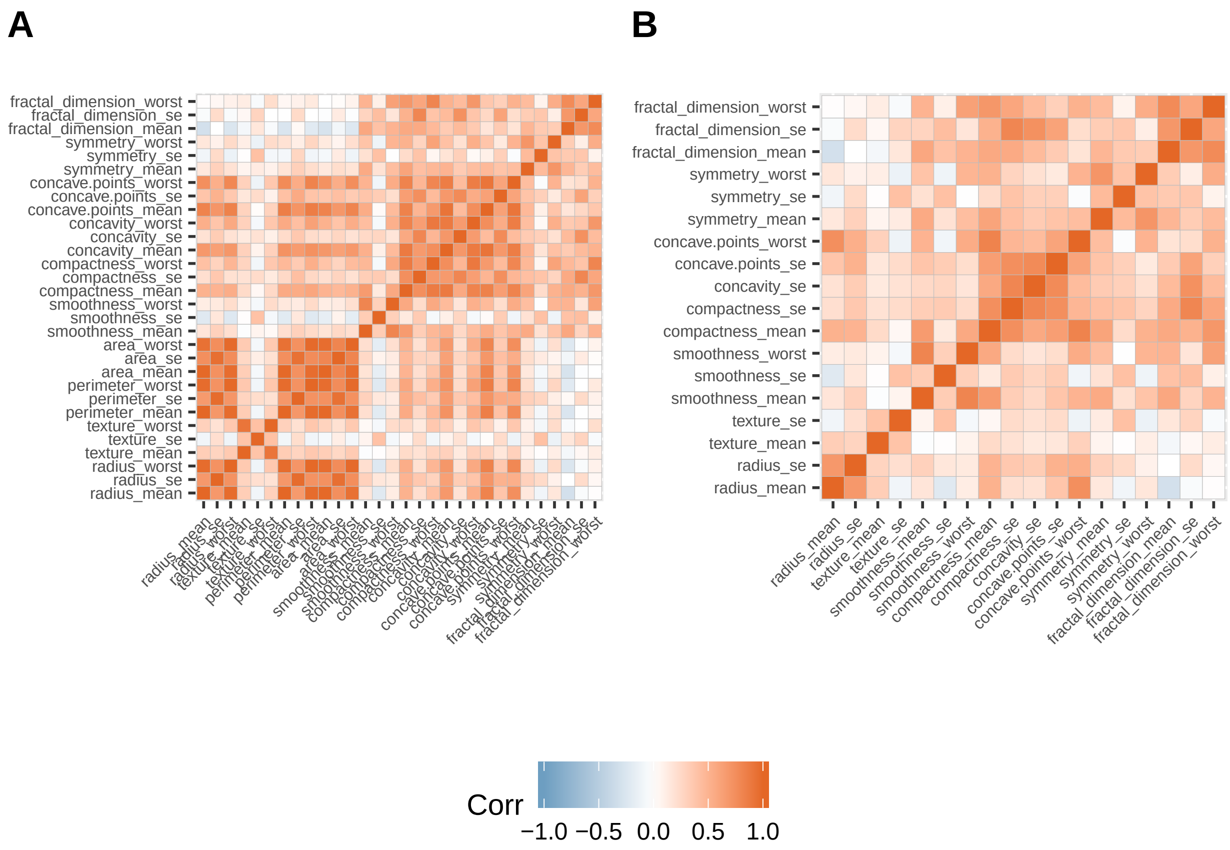
The data *breast-cancer.csv* have 569 row and 33 columns. The first column **ID** lables individual breast tissue images; The second column **Diagnonsis** indentifies if the image is coming from cancer tissue or benign cases (M=malignant, B = benign). There are 357 benign and 212 malignant cases. The other 30 columns correspond to mean, standard deviation and the largest values (points on the tails) of the distributions of the following 10 features computed for the cellnuclei.

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness ($perimeter^2/area$ - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

Variable selection

There exists multicollinearity within the dataset (**Fig. 1A**), which will casue the correlation matrix to be singular and irreversible. We proposed a method to reduce this multicollinearity by deleting some variables. The chosen criterion is based on both correlation coefficient and eigen value of correlation matrix, that is, the column with high correlation with other columns (e.g., larger than 0.7) and low eigen value (e.g., less than 0.01) will be deleted.

Figure 1: multicollinearity plot of the dataset



Final input of model

There were 18 variables left as final input predictors (except for **Lasso** model), **Fig. 1B** shows the correlation matrix after deletion.

Method

Logistic Regression

Let y be the vector n response random variable, X denote the $n \times p$ design matrix(letX_i denote the ith row) and β denote the $p \times 1$ coefficient.

$$y \sim Bin(n, p)$$
$$g(E(y)) = X\beta$$

Given the link function as

$$g(\eta) = log\frac{\eta}{1-\eta}$$

Modified Newton-Raphson Algorithm with Step Halving

The likelihood of logistic regression is:

$$L(\beta; X, y) = \prod_{i=1}^{n}\{(\frac{\exp(X_i\beta)}{1+\exp(X_i\beta)})^{y_i}(\frac{1}{1+\exp(X_i\beta)})^{1-y_i}\}$$

Maximizing the likelihood is equivalent to maximizing the log likelihood:

$$f(\beta) = \sum_{i=1}^{n}\{y_i(X_i\beta) - \log(1+\exp(X_i\beta))\}$$
$$=< X\beta, Y > - \sum_{i=1}^{n}\log(1+\exp(X_i\beta))$$

Let p, a vector of n denote $p = \frac{\exp(X\beta)}{1+\exp(X\beta)}$. The gradient of this function is:

$$\nabla f(\beta) = X^T(y-p)$$

The Hessian is given by:

$$\nabla^2 f(\beta) = -X^T W X$$

where $W = diag(p_i(1-p_i)), i = 1, \cdots, n$ Hessian matrix is negative definite, well behaved.

The above can be regarded as a method to search for solutions to the system of nonlinear equations $\nabla f(\beta) = 0$.

At each step, given the current point β_i, the gradient $\nabla f(\beta)$ for β near β_i may be approximated by

$$\nabla f(\beta_i) + \nabla^2 f(\beta_i)(\beta - \beta_i)$$

which defines the plane tangent to $\nabla f(\beta)$ at β_i. The next step in the algorithm is determined by solving the system of linear equations

$$\nabla f(\beta_i) + \nabla^2 f(\beta_i)(\beta - \beta_i) = 0$$

and the next "current point" is set to be the solution

$$\beta_{i+1}(\lambda) = \beta_i - \lambda[\nabla^2 f(\beta_i)]^{-1}\nabla f(\beta_i)$$

Then compute the full Newton-Raphson step, corresponding to $\lambda = 1$. If $f(\theta_{i+1}(1)) \geq f(\theta_i)$, then set $\theta_{i+1} = \theta_{i+1}(1)$ and go on to the next iteration. Otherwise, search for a value $\lambda \in (0,1)$ for which $f(\theta_{i+1}(\lambda)) \geq f(\theta_i)$, set $\theta_{i+1} = \theta_{i+1}(\lambda)$ and go on to the next iteration.

The algorithm continues like this iteratively choosing a sequence of points $\beta_0, \beta_1, ..., \beta_i, ...$ until convergence is achieved.

Newton Raphson with Gradient Decent Update

For Newton's method with a large p, the computational burden in calculating the inverse of the Hessian Matrix $[\nabla^2 f(\beta_i)]^{-1}$ increases quickly with p. One can update

$$\beta_{i+1} = \beta_i + I_{p\times p}\nabla f(\beta_i)$$

This is easy to compute, but could slow in convergence.

Coordinate-wise descent LASSO with weighted updates

Regularization is the common variable selection approaches for high-dimensional covariates. The best known Regularization is called LASSO. In linear regression, LASSO minimize

$$f(\beta) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{i,j}\beta_j)^2 + \gamma\sum_{j=1}^{p}|\beta_j|$$

for some $\gamma \geq 0$. Here the $x_{i,j}$ are standardized so that $\sum_i x_{i,j}/n = 0$ and $\sum_i x_{i,j}^2/n = 0$

With a single predictor, the LASSO solution is very simple

$$\hat{\beta}^{lasso}(\gamma) = S(\hat{\beta}, \gamma) = sign(\hat{\beta})(|\hat{\beta}| - \gamma)_+$$

When p is large, the optimization could be challenging, so we could use coordinate-wise descent LASSO with objective function

$$f(\beta_j) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \sum_{k\neq j} x_{i,k}\tilde{\beta}_k - x_{i,j}\beta_j)^2 + \gamma\sum_{k\neq j}|\tilde{\beta}_k| + \gamma|\beta_j|$$

Minimize $f(\beta_j)$ w.r.t. β_j while having $\tilde{\beta}_k$ fixed, we have

$$\tilde{\beta}_j^{lasso}(\gamma) \leftarrow S(\sum_{i=1}^{n} x_{i,j}(y_i - \tilde{y}_i^{(-j)}), \gamma)$$

where $\tilde{y}_i^{(-j)} = \sum_{k\neq j} x_{i,k}\tilde{\beta}_k$.

A weight ω_i is then associated with each observation. In this case the update becomes only slightly more complicated:

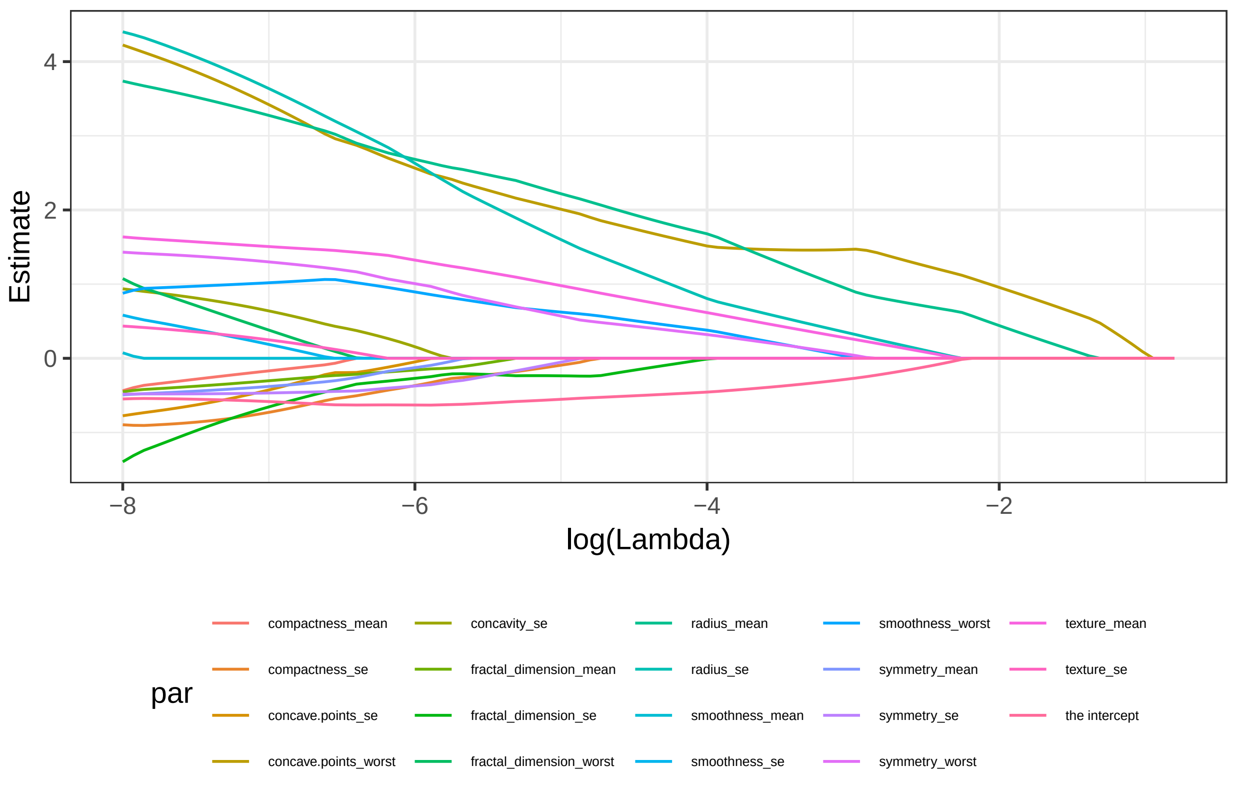
$$\tilde{\beta}_j^{lasso}(\gamma) \leftarrow \frac{S(\sum_{i=1}^{n}\omega_i x_{i,j}(y_i - \tilde{y}_i^{(-j)}), \gamma)}{\sum_i \omega_i x_{i,j}^2}$$

Results

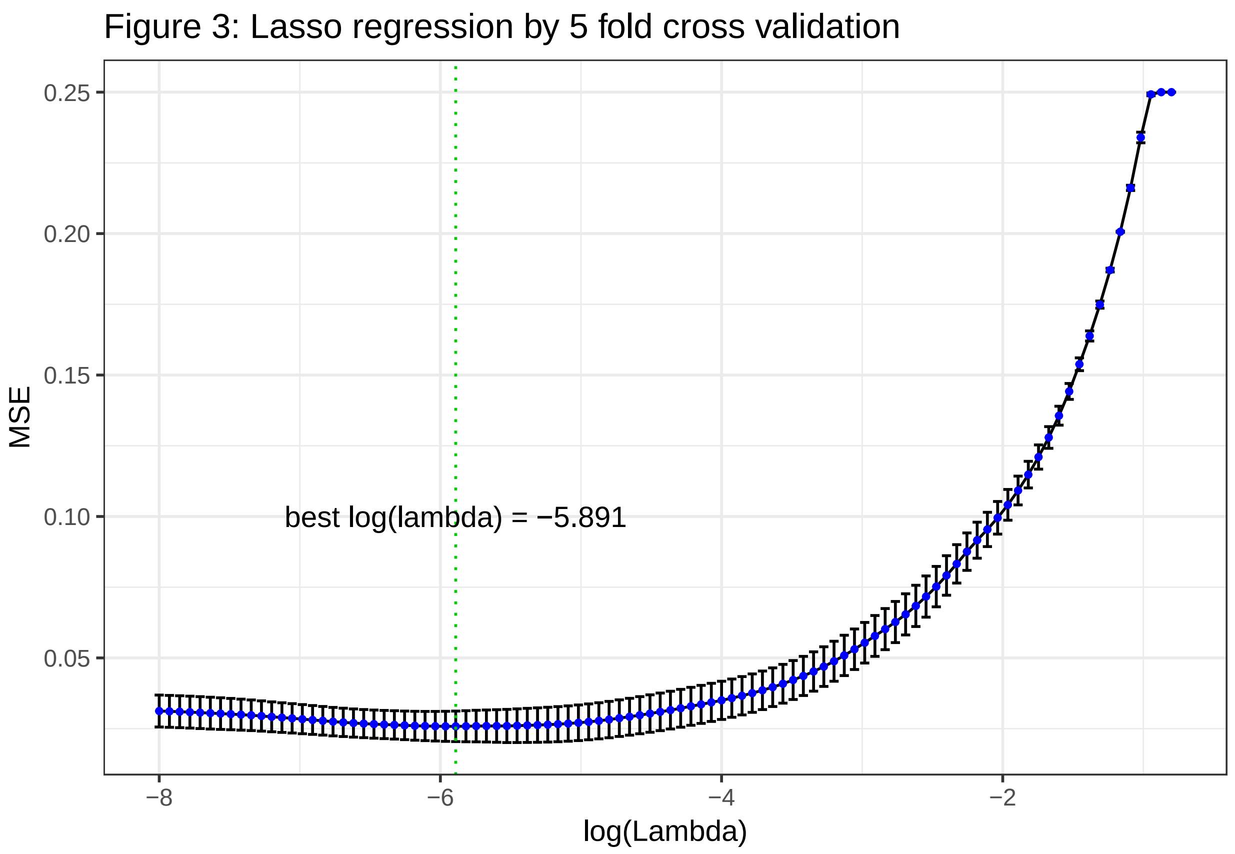
Estimation path for Lasso

Fig. 2 shows us that as the λ increases, all the variable parameters shrink accordingly. When $\lambda = 0$, the result is the same as Least Square method and when λ is too large, all the parameters shrink to 0.

Figure 2: A path of solutions with a sequence of descending lambda's



Cross validation for lasso



When using 5-fold cross-validation, we calculate the mean squared error (MSE) and its $\pm 1\sigma^2$ interval as the output. **Fig. 3** shows us that the *lambda* with the lowest MSE is 0.00276.

Model comparison

We finally compared these four methods: Classic Logistic Regression (GLM package), Modified Newton-Raphson Algorithm with Step Halving (Newton Raphson), Newton Raphson with Gradient Decent Update (Gradient Decent) and Coordinate-wise descent LASSO with weighted updates (Logistic Lasso). **Table 1** shows the MSE and iteration time they need to converge: Newton Raphson method needs the lowest iteration times while Gradient Decent method needs more than 1000 times to converge. **Table 2** shows the final parameter estimation calculated by these four methods. Newton Raphson got the exact same results as the build-in GLM, which proved the validity of our algorithm. And the Gradient Decent result is similar to Newton Raphson. With the best λ, Lasso results in turning 6 parameters to 0.

Table 1: The comparison of performance for estimation algorithms and models

	GLM package	Newton Raphson	Gradient Decent	Logistic Lasso
iteration times	NA	12	1001	100
MSE	0.02	0.02	0.02	0.02

[a] Dataset: Breast Cancer Diagnosis

Table 2: The comparison of performance for estimation algorithms and models

	GLM package	Newton Raphson	Gradient Decent	Logistic Lasso
radius_mean	4.43	4.43	3.18	2.63
texture_mean	1.89	1.89	1.34	1.29
smoothness_mean	0.78	0.78	0.47	0.00
compactness_mean	-1.14	-1.14	-0.59	0.00
symmetry_mean	-0.63	-0.63	-0.44	-0.10
fractal_dimension_mean	-0.66	-0.66	-0.72	-0.14
radius_se	5.13	5.13	3.28	2.50
texture_se	0.59	0.59	0.46	0.00
smoothness_se	1.10	1.10	0.77	0.00
compactness_se	-0.80	-0.80	-0.68	-0.33
concavity_se	1.24	1.24	0.88	0.08
concave.points_se	-1.11	-1.11	-0.80	0.00
symmetry_se	-0.53	-0.53	-0.39	-0.36
fractal_dimension_se	-2.73	-2.73	-1.55	-0.25
smoothness_worst	0.31	0.31	0.31	0.86
concave.points_worst	5.13	5.13	3.65	2.48
symmetry_worst	1.60	1.60	1.28	0.97
fractal_dimension_worst	2.19	2.19	1.41	0.00
intercept	-0.62	-0.62	-0.71	-0.63

[a] Dataset: Breast Cancer Diagnosis

Conclusions

References

1 Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. "Regularization paths for generalized linear models via coordinate descent." Journal of statistical software 33.1 (2010): 1.

2 Friedman, Jerome, et al. "Pathwise coordinate optimization." The annals of applied statistics 1.2 (2007): 302-332.

Appendix A

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, comment = "")
library(tidyverse)
library(Hmisc)#for errvar
library(ggcorrplot) # for correlation heatmap
library(ggpubr) # common legend
library(matrixcalc) #is.negative.def
library(kableExtra) # table
library(glmnet)
set.seed(99999)
options(knitr.table.format = "latex")
theme_set(theme_bw())
###### data manipulation
```

```r
mydata <- read.csv("breast-cancer-1.csv")
n <- dim(mydata)[1]
p <- dim(mydata)[2]
list <- c(3:(p - 1))
namesX <- names(mydata)[-c(1,2,p)]
# standardize
dataX <- do.call(cbind, lapply(list, function(x) (mydata[,x] - mean(mydata[,x]))/sd(mydata[,x])))
# design matrix
X <- data.frame(dataX) %>%
  mutate(., intercept = 1)
# response
resp <- as.vector(ifelse(mydata[,2] == "M", 1, 0))
###### plot to check collinearity
colnames(dataX) <- namesX
colinearity.plot <- function(data){
  data.frame(data) %>%
  select(starts_with("radius"),
         starts_with("texture"),
         starts_with("perimeter"),
         starts_with("area"),
         starts_with("smooth"),
         starts_with("compact"),
         starts_with("concavity"),
         starts_with("concave"),
         starts_with("symmetry"),
         starts_with("fractal")) %>%
  cor() %>%
  ggcorrplot(.,ggtheme = ggplot2::theme_gray,
             colors = c("#6D9EC1", "white", "#E46726"),
             tl.cex = 6)}
g1 <- colinearity.plot(dataX)

###### variable selection
eig <- eigen(cor(dataX))$values
# found values very close to 0, multicolinearity exists
# function: find the maximum correlation between i and j, at least 0.5
max.corr <- function(data){
  len = dim(data)[2]
  a <- 0.5
  for (i in 1:(len-1)) {
    for (j in (i+1):len) {
      if (abs(cor(data[,i],data[,j])) > a) a <- cor(data[,i],data[,j])
    }
  }
  return(round(a,3))
}
# function: update dataset according to several rules: eigenvalues and corr
selection <- function(data, eigen.tol, corr.tol) {
  while (min(eigen(cor(data))$values) <= eigen.tol & max.corr(data) >= corr.tol) {
       temp <- data
       data <- temp[,-(which(round(abs(cor(temp)),3) == max.corr(temp), arr.ind = TRUE)[1,1])]
      }
  return(data)}
```

```r
newdataX <- selection(dataX, eigen.tol = 2e-2, corr.tol = 0.7)
g2 <- colinearity.plot(newdataX)
eigpost <- eigen(cor(newdataX))$values
# look at the difference of the deleted colums
delnames <- setdiff(colnames(dataX),colnames(newdataX))
ggarrange(g1,g2, ncol=2, nrow=1, common.legend = TRUE, legend = "bottom", labels = "AUTO")
###### 0.logistic regression
# logfit.0 <- glm(resp~dataX, family = binomial(link = "logit"))
# algorithm didn't converge without delete colinearity
logdata <- cbind.data.frame(resp,newdataX)
logfit.1 <- glm(resp~., family = binomial(link = "logit"),data = logdata)
logit.beta <- coef(logfit.1)
##### 1. classical newton raphson
newX <- data.frame(newdataX) %>%
  mutate(., intercept = 1)
newdat <- list(y = resp, X = as.matrix(newX))
# function: calcualte loglik, gradient, hessian
logisticstuff <- function(dat, betavec){
  u <- dat$X %*% betavec
  expu <- exp(u)
  loglik <- t(u) %*% dat$y - sum((log(1 + expu))) # Log-likelihood at betavec
  prob <- expu / (1 + expu) # P(Y_i=1|x_i)
  grad <- t(dat$X) %*% (dat$y - prob)
  Hess <- -t(dat$X) %*% diag(as.vector(prob*(1 - prob))) %*% dat$X # Hessian at betavec
  return(list(loglik = loglik, grad = grad, Hess = Hess))}

NewtonRaphson <- function(dat, func, start, tol=1e-10, maxiter =200){
  i <- 0
  cur <- start
  stuff <- func(dat, cur)
  res <- c(0, stuff$loglik, cur)
  prevloglik <- -Inf
  while (i < maxiter && abs(stuff$loglik - prevloglik) > tol && stuff$loglik > -Inf) {
    i <- i + 1
    prevloglik <- stuff$loglik
    prev <- cur
    lambda = 0
    cur <- prev - ((1/(2^lambda)) * solve(stuff$Hess)) %*% stuff$grad
    while (func(dat, cur)$loglik < prevloglik) { # step-halving
      lambda = lambda + 1
      cur <- prev - ((1/(2^lambda)) * solve(stuff$Hess)) %*% stuff$grad
    }
    stuff <- func(dat, cur)
    res <- rbind(res, c(i, stuff$loglik, cur))}

  return(res)
}
newres <- NewtonRaphson(newdat,logisticstuff, start = rep(0, dim(newdat$X)[2]))
# check convergence
check <- tail(newres)[,1:2]
newton.beta <- newres[nrow(newres),3:dim(newres)[2]]
##### 2. gradient descent
gradient <- function(dat, func, start, tol=1e-10, maxiter = 200){
  i <- 0
  cur <- start
  pp <- length(start)
  stuff <- func(dat, cur)
  hessinversed <- solve(t(dat$X) %*% (dat$X))# double check
  res <- c(0, stuff$loglik, cur)
  prevloglik <- -Inf
  while (i < maxiter && abs(stuff$loglik - prevloglik) > tol && stuff$loglik > -Inf) {
    i <- i + 1
    prevloglik <- stuff$loglik
    prev <- cur
    cur <- prev + hessinversed %*% (stuff$grad)
    stuff <- func(dat, cur)
    res <- rbind(res, c(i, stuff$loglik, cur))}
  return(res)
}
gradres <- gradient(newdat, logisticstuff, start = rep(0, dim(newdat$X)[2]), maxiter = 1000)
# check convergence
check <- tail(gradres)[,1:2]
grad.beta <- gradres[nrow(gradres),3:dim(gradres)[2]]
##### 3. coordinate-wise logistic lasso

sfun <- function(beta,lambda) sign(beta) * max(abs(beta)-lambda, 0)
coordinatelasso <- function(lambda, dat, s, tol=1e-10, maxiter = 200){
  i <- 0
  pp <- length(s)
  n <- length(dat$y)
  betavec <- s
  loglik <- 1e6
  res <- c(0, loglik, betavec)
  prevloglik <- Inf # To make sure it iterates
  while (i < maxiter && abs(loglik - prevloglik) > tol && loglik < Inf) {
    i <- i + 1
    prevloglik <- loglik
    for (j in 1:pp) {
      u <- dat$X %*% betavec
      expu <- exp(u)
      prob <- expu/(expu+1)
      w <- prob*(1-prob) # weighted
      # avoid coeffcients diverging in order to achieve fitted  probabilities of 0 or 1.
      w <- ifelse(abs(w-0) < 1e-5, 1e-5, w)
      z <- u + (dat$y-prob)/w
      # calculate noj
      znoj <- dat$X[,-j] %*% betavec[-j]
      # revise the formula to be z
      betavec[j] <- sfun(mean(w*(dat$X[,j])*(z - znoj)), lambda)/(mean(w*dat$X[,j]*dat$X[,j]))
    }
    loglik <- sum(w*(z-dat$X %*% betavec)^2)/(2*n) + lambda * sum(abs(betavec))
    res <- rbind(res, c(i, loglik, betavec))}
  return(res)
}
corres <- coordinatelasso(lambda = exp(-8e-1), newdat, s = rep(0, dim(newdat$X)[2]) ,maxiter = 2000)
check <- tail(corres)[,1:2]
```

```
cor.beta <- corres[nrow(corres),3:dim(corres)[2]]
logmod <- glmnet(newdat$X[,-dim(newdat$X)[2]], y=newdat$y, alpha=1, family="binomial",lambda=1e-2)
# check: coef.glmmet(logmod)
logmod <- cv.glmnet(newdat$X[,-dim(newdat$X)[2]], y=newdat$y, alpha=1, family="binomial",type.measure="
# check: plot(logmod)
# check: logmod$lambda.min
# impletement the pathwise coordinatewise optimization algorithm to obtain a path of solutions
path <- function(inputx,inputy,grid){
  start <- rep(0, dim(inputx)[2])
  betas <- NULL
  for (x in 1:100) {
  cv.errors <- vector()
    cor.result <- coordinatelasso(lambda = grid[x],
                                  dat = list(X=as.matrix(inputx),y=inputy),
                                  s= start)
    lasbeta <- cor.result[nrow(cor.result),3:dim(cor.result)[2]]
    start <- lasbeta
    betas <- rbind(betas,c(lasbeta))
  }
  return(data.frame(cbind(grid,betas)))
}
path.out <- path(newX,resp,grid=exp(seq(-8e-1,-8, length=100)))
colnames(path.out) <- c("grid",colnames(newdataX),"the intercept")
# plot a path of solutions
path.plot <- path.out %>%
  gather(key = par, value = estimate, c(2:dim(path.out)[2])) %>%
  ggplot(aes(x = log(grid), y = estimate, group = par, col = par)) +
  geom_line()+
  ggtitle("Figure 2: A path of solutions with a sequence of descending lambda's") +
  xlab("log(Lambda)") +
  ylab("Estimate") +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 5))
path.plot
##### 5-fold cross-validation and pathwise coordinatewise optimization algorithm
# when lambda = 0.5, all the betas go to 0
cvresult <- function(inputx,inputy,grid,K){
  n <- dim(inputx)[1]
  folds <- sample(1:K, n, replace=TRUE)
  start <- rep(0, dim(inputx)[2])
  cv.error <- vector()
  cv.se <- vector()
  for (x in 1:length(grid)) {
  cv.errors <- vector()
  for (i in 1:K){
    cor.result <- coordinatelasso(lambda = grid[x],
                                  dat = list(X=as.matrix(inputx[folds!=i,]),y=inputy[folds!=i]),
                                  s = start)
    lasbeta <- cor.result[nrow(cor.result),3:dim(cor.result)[2]]
    u <- as.matrix(inputx[folds == i,]) %*% lasbeta
    expu <- exp(u)
    prob <- expu / (1 + expu)
    y <- as.vector(inputy[folds==i])
    cv.errors[i] = mean((y-prob)^2) #MSE
    start <- lasbeta
  }
  cv.error[x] <- mean(cv.errors)
  cv.se[x] <- sqrt(var(cv.errors)/K)
  }
  return(cbind(grid,cv.error,cv.se))
}

result <- cvresult(newX,resp,grid=exp(seq(-8e-1,-8, length=100)),K=5)
# result <- cvresult(X,resp,grid=seq(0.5, 1e-2, length=100),K=5)
best.lambda <- result[which.min(result[,2]),1]
# need rewrite
finlasso <- as.matrix(path(newX,resp,grid=exp(seq(-8e-1,log(best.lambda), length=100))))
lasso.beta <- finlasso[nrow(finlasso),2:dim(finlasso)[2]]

# plot for cross validation
result <- data.frame(result)
cv.plot <-
    ggplot(result, aes(x=log(result$grid), y=result$cv.error)) +
    geom_errorbar(aes(ymin=result$cv.error-result$cv.se, ymax=result$cv.error+result$cv.se),
                  colour=1) +
    geom_line() +
    geom_point(size=0.8,colour = 4) +
    ggtitle("Figure 3: Lasso regression by 5 fold cross validation")+
    xlab("log(Lambda)") + ylab("MSE") +
    geom_vline(xintercept = log(best.lambda),col=3,lty=3) +
    annotate("text", log(best.lambda), 0.1, label = paste("best log(lambda) = ", round(log(best.lambda)

cv.plot
######## compare prediction performance of all results
pred.fun <- function(outcome,input, beta){
    u <- as.matrix(input) %*% beta
    expu <- exp(u)
    prob <- expu / (1 + expu)
    pred.error = mean((as.vector(outcome)-prob)^2)
    return(pred.error)
}

# logistic regression by GLM
log.beta <- c(logit.beta[2:length(logit.beta)],logit.beta[1])
pred <- predict(logfit.1)
log.pred <- mean((resp-exp(pred)/(1+exp(pred)))^2) # abs(mean(logfit.1$residuals))
# newton's method
newton.ite <- nrow(newres)
newton.beta <- newres[nrow(newres),3:dim(newres)[2]]
newton.pred <- pred.fun(resp,newX,newton.beta)
# gradient decent
grad.ite <- nrow(gradres)
grad.beta <- gradres[nrow(gradres),3:dim(gradres)[2]]
grad.pred <- pred.fun(resp,newX,grad.beta)
# lasso logistic
lasso.ite <- nrow(finlasso)
lasso.beta <- lasso.beta
lasso.pred <-  pred.fun(resp,newX,lasso.beta)

beta.res <- round(as.matrix(rbind(log.beta,newton.beta,grad.beta,lasso.beta)),2)
colnames(beta.res) <- colnames(newX)
rownames(beta.res) <- c("GLM package","Newton Raphson","Gradient Decent","Logistic Lasso")
perf.res <- matrix(rep(NA),ncol = 2, nrow = 4)
colnames(perf.res) <- c("iteration times","MSE")
rownames(perf.res) <- c("GLM package","Newton Raphson","Gradient Decent","Logistic Lasso")
perf.res[1,1] <- "NA"
perf.res[1,2] <- round(log.pred ,2)
perf.res[2,1] <- newton.ite
perf.res[2,2] <- round(newton.pred,2)
perf.res[3,1] <- grad.ite
perf.res[3,2] <- round(grad.pred,2)
perf.res[4,1] <- lasso.ite
perf.res[4,2] <- round(lasso.pred,2)

# output-performace
kable(t(perf.res), "latex", caption = "The comparison of performance for estimation algorithms and mode
  kable_styling(latex_options = c("hold_position", "scale_down")) %>%
  add_footnote(c("Dataset: Breast Cancer Diagnosis"),
               notation = "alphabet")

# output-beta
kable(t(beta.res), "latex", caption = "The comparison of performance for estimation algorithms and mode
  kable_styling(latex_options = c("hold_position", "scale_down")) %>%
  add_footnote(c("Dataset: Breast Cancer Diagnosis"),
               notation = "alphabet")
```