Term Project

Statistical Asymptotic Theory in Big Data
STA 7734

Poorna Sandamini Senaratne

Contents

1 Introduction

1.1 Goodness-of-fit tests

Goodness-of-fit tests are methods used to determine how well an observed set of data fits a particular distribution or model. They compare the observed values with the expected values or the fitted values.

A goodness-of-fit statistic tests the following hypothesis,

$$H_0 : \textit{the model fits} \text{ vs } H_1 : \textit{the model does not fit}$$

There are several types of goodness-of-fit tests that can be used in different types of scenarios. Given below are some examples of goodness-of-fit tests that are commonly used.

1. Chi-Square test: tests the validity of a claim made about a population based on a random sample.
2. Kolmogorov-Smirnov test: determines whether a sample is from a specific distribution within a population.
3. Shapiro-Wilk test: determines if a sample follows a normal distribution.
4. Hosmer-Lemeshow test: compares the expected frequencies of a binary outcome with the observed frequencies of that outcome in different groups or intervals.
5. AIC : measures the relative quality of a statistical model for a given set of data.
6. BIC: used for model selection among a finite set of models.

1.2 High dimensional setting

All the scenarios explained above fits well when there is low-dimensional data. They mostly rely on settings like asymptotic linearity and normality of the maximum likelihood estimator. Problem arises when there is a high-dimensional data setting and how to apply the goodness-of-fit tests in that respect. Because the methods explained might fail to assess the goodness-of-fit when the number of covariates increases.

This problem has motivated people to come up with a new method to assess the goodness-of-fit of a model when there is high dimensional data. Specifically, a method to detect misspecification in the fit of a generalized linear model with a high dimensional setting.

The new method introduces a framework for analyzing data in the context of generalized linear models. These models involve feature vectors $x_i \in \mathbb{R}^p$ and univariate responses $Y_i \in Y \subseteq \mathbb{R}$. The design matrix is denoted as $X = [x_1, \ldots, x_n]^T = [X_1, \ldots, X_p] = (X_{ij})_{1 \leq i \leq n, 1 \leq j \leq p}$, and the response vector is $Y = (Y_1, \ldots, Y_n)^T$.

The focus is on a generalized linear model where Y_i are conditionally independent given X, and the distribution of Y_i depends on X through the linear combination $x_i^T\beta_0$. Here, $E(Y_i|x_i = x) =: m_0(x) = \mu(x^T\beta_0)$, where $\mu(\cdot)$ is a known inverse link function, and β_0 is an unknown coefficient vector. Additionally, it is assumed that $Var(Y_i|x_i = x) = \mu'(x^T\beta_0)$, a structure often seen in generalized linear models from exponential families with canonical link functions.

The interest lies in detecting misspecification in the conditional mean function. In low-dimensional settings, misspecification occurs when there is no $\beta_0 \in \mathbb{R}^p$ such that $m_0(x) = \mu(x^T\beta_0)$. In high-dimensional settings ($p \geq n$), the concept is more complex. Despite fixed design points $x_1, \ldots, x_n$, consistent estimation of β_0 is impossible without additional structural assumptions. The paper adopts sparsity as an assumption, exploring whether a sparse model fits the observations well or if a sparse non-linear model is more appropriate.

In a high-dimensional context, the paper addresses the question of whether a sparse model, subject to sparsity assumptions, adequately captures the underlying data structure. The challenge arises from the inability to consistently estimate β_0 without further assumptions. The discussion encompasses various types of misspecification, with a focus on sparse models and the potential omission of nonlinear terms, such as quadratic effects or interaction terms.

Some examples of generalized linear models within the introduced framework include logistic regression, Poisson regression, robust regression (Huber loss, Cauchy loss), and linear regression.

As such we will be looking into the application of goodness-of-fit test in high dimensional data with respect to logistic regression.

2 Data Collection and Preparation

2.1 Data Collection

Dataset have been taken from the UCI Machine Learning Repository. The dataset was named as the Toxicity Dataset.

Observations include 171 molecules designed for functional domains of a core clock protein, CRY1, responsible for generating circadian rhythms.CRY1, the circadian cryptochrome, is a pro-tumorigenic factor that rhythmically modulates DNA repair. The dataset consists of 1203 variables. That is it contains a complete set of molecular descriptors. The dependent variable is a binary variable which classifies whether the molecule is toxic or non-toxic. 33% were toxic molecules, while 67% were non-toxic molecules.

2.2 Data Preparation

There were no missing values in the datatset. However, several data transformations were conducted so that the dataset could be used in the further steps of the analysis.

First quantile normalization on the data matrix was performed. It calculates the ranks of the columns of the data matrix and transforms the values to quantiles by dividing by (n + 1), where n is the number of observations. Then the quantiles were matched to the corresponding quantiles of a standard normal distribution. As the next step the covariance matrix was calculated. This was done, after quantile normalization and ranking on the original matrix. In order to decompose the covariance matrix into a lower triangular matrix Cholesky decomposition was utilized. Later on, the pivot vector from the Cholesky decomposition was extracted and the columns of the covariance matrix were reoreded according to the pivot vector. Also, the columns of the original matrix were also ordered based on their variances in decreasing order.

These operations were aimed to transform the data and its covariance structure, preparing it for further analyses or modeling.

3 Model Discussion

3.1 Theory

The paper discusses the following asymptotic theory and implements it in the data implementation. The term project is alos built on this theory.

Condition 3. Assume that $\mathbb{E}\left(Y_i \mid x_i = x\right) = \mu\left(f_0(x)\right)$, the inverse link function $u \mapsto \mu(u)$ is differentiable, $u \mapsto \mu'(u)$ is Lipschitz with constant L, and $\mu'(u) > 0$ for all $u \in \mathbb{R}$. Suppose moreover that the weights satisfy $\min_i D_{\beta_0,ii} \geq d_{\min}$ for some constant $d_{\min} > 0$. Assume that $\mathbb{E}\left(\left|Y_i - \mu\left(f_0\left(x_i\right)\right)\right|^3 / D_{Y,ii}^3 \mid X_i\right) \leq C_\varepsilon$ for a constant $C_\varepsilon > 0$, where we denote $D_Y^2 := \operatorname{cov}(Y \mid X)$. Let $\max_{i=1,\ldots,n} \|x_i\|_\infty \leq K_X$ for some $K_X \geq 1$ and assume that $12 d_{\min}^{-2} L K_X s\lambda \leq 1$ and $\mid D_{Y,ii}^2 D_{\beta_0,ii}^{-2} - 1 \mid \leq 2 d_{\min}^{-2} L K_X s\lambda$.

Theorem 3. Consider Algorithm 1 with tuning parameters $\lambda, \lambda_A, \lambda_{\text{sq}}$. Assume Condition 3, that $\hat{\beta}_A \in \Theta\left(\lambda, \beta_0, X_A\right)$ and let

$$\delta := \mathbb{P}\left(\hat{\beta} \notin \Theta\left(\lambda, \beta_0, X\right) \mid X\right).$$

Then there exists a constant $C > 0$ such that, whenever Z_A is such that $\hat{f}(X) \neq X\hat{\beta}_{\text{sq}}$, we have for any $z \in \mathbb{R}$ that

$$\left|\mathbb{P}\left(T - \Delta < z \mid Z_A\right) - \Phi(z)\right| \leq \delta + C\left\{\lambda_{\text{sq}}\sqrt{n} s\lambda + \|\hat{w}_A\|_\infty s\left(\lambda^2 + \lambda_A^2\right) n + K_X s\lambda_A + \|\hat{w}_A\|_\infty\right\},$$

where

$$\Delta := \hat{w}_\Lambda^T\left\{\mu\left(f_0(X)\right) - \mu\left(X\beta_0\right)\right\}.$$

3.1.1 Logistic Regression approach

The logistic regression model is considered where the response variable Y takes values in $\{0, 1\}$, and the conditional distribution is assumed to follow a Bernoulli distribution with parameter $\pi_0(x)$.

The inverse link function is denoted as $\mu(u) = 1/(1 + e^{-u})$, and the function $f_0(x)$ is defined as

$$f_0(x) := \log\left(\frac{\pi_0(x)}{1 - \pi_0(x)}\right)$$

This formulation allows expressing the conditional expectation $E(Y_i|x_i = x)$ as $\pi_0(x) = \mu(f_0(x))$.

The logistic regression estimator is introduced as

$$\hat{\beta} := \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \left\{-Y_i x_i^T \beta + d(x_i^T \beta) + \lambda \|\beta\|_1\right\}$$

where $d(\xi) := \log(1 + e^{\xi})$.

The best approximation β_0 obtained by a Generalized Linear Model (GLM) is considered.

Corollary 1 is presented as a result of combining Theorem 3 with existing results on L_1-penalized logistic regression. The conditions of Lemma 6 are assumed, ensuring that the penalized estimator is sufficiently close to β_0 with high probability.

Corollary 1 states, under certain conditions, the existence of a constant $C > 0$ such that the difference between the probability $P(T - \Delta < z|Z_A)$ and the standard normal cumulative distribution function $\Phi(z)$ is bounded. The term Δ is defined in terms of the difference between the true and estimated logistic regression functions, and the bound involves factors such as the number of non-zero coefficients (s), the logarithm of the dimensionality ($\log(2p)$), and the L_∞ norm of the estimated weights ($\|\hat{w}_A\|_\infty$).

3.2 Models used

1. Lasso (L1 Regularization)

The Lasso method is a regularization technique that adds a penalty term proportional to the absolute values of the coefficients to the ordinary least squares (OLS) cost function. It encourages sparsity in the model by shrinking some coefficients to exactly zero.

$$\text{argmin}_\beta \left(\|Y - X\beta\|_2^2 + \lambda\|\beta\|_1\right)$$

2. Ridge (L2 Regularization)

Ridge regression is a regularization method that adds a penalty term proportional to the square of the coefficients to the OLS cost function. It helps prevent overfitting and stabilizes the estimates, but unlike Lasso, it does not enforce sparsity.

$$\text{argmin}_\beta \left(\|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2\right)$$

3. Adaptive Lasso

Adaptive Lasso is an extension of Lasso that applies different penalty weights to each coefficient. The weights are usually inversely proportional to the absolute values of the estimated coefficients from an initial model. This approach encourages more shrinkage for less important variables.

$$\text{argmin}_\beta \left(\|Y - X\beta\|_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j| \right)$$

4. Debiased Lasso

Debiased Lasso is designed to mitigate the bias introduced by Lasso. It adjusts the Lasso estimates to account for the bias induced by the variable selection property of Lasso. This can improve the accuracy of coefficient estimates.

$$\hat{\beta}^{\text{deb}} = \hat{\beta}^{\text{lasso}} + (X^T X)^{-1} X^T (Y - X\hat{\beta}^{\text{lasso}})$$

Feel free to use these LaTeX expressions in your documents or presentations.

4 Methods and Results

4.1 Simulated example

4.1.1 Method

We will be conducting a hypothesis test, $H_0 : \sigma = 0$ at significance level 0.05 for different values of ρ and σ. In this context, the logistic regression models are considered and analyzed to check how they handle misspecification. There are two types of misspecification we're exploring: one involving a pure quadratic effect and the other involving an interaction term.

The log-odds function we're working with is defined as $f_0(u) = u^T\beta_0 + \sigma g(u_1, \ldots, u_p)$, where $\beta_0 = where\beta_0 = (-1, 1, -1, 1, -1, 0, \ldots, 0) \in \mathbb{R}^p$. The function g(u) introduces the misspecification, and we have two scenarios: (a) $(g(u) = u_1u_2)$ or (b) $g(u) = u_1^{1/2}$. The parameter σ tunes how much misspecification we want, ranging from 0 to 3.

The observation vectors x_i is independent and follow a Gaussian distribution $N_p(0, \Sigma_0)$. This matrix Σ_0 is a Toeplitz matrix, which depends on a range of correlations $\rho \in \{0.4, 0.6, 0.8\}$.

In the setup, we consider p = 500 variables and N=800 observation. To perform the GRP-test, the sample needs to be split. We're doing a 50% split, meaning the auxiliary sample has a size of nA = 400. For this test, we're using a random forest as the method to predict residuals.

In the context of a high-dimensional setting, direct comparisons for the proposed GRP-test are challenging. To establish a theoretical benchmark, an oracle GRP-test applied to a reduced design

matrix that includes only variables in the active set $S = \{1, \ldots, 5\}$, effectively reducing the problem to a low-dimensional one. From the results it could be seen that the GRP-test effectively controls the type I error. Additionally, it exhibits a relatively modest decrease in power when compared to the oracle GRP-test.

4.1.2 Results

Interaction effect

GRP-test Results

	$\sigma = 0$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$
$\rho = 0.25$	0.02	0.04	0.06	0.1
$\rho = 0.5$	0.02	0.08	0.26	0.46
$\rho = 0.75$	0.06	0.34	0.76	0.78

Benchmark Results

	$\sigma = 0$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$
$\rho = 0.25$	0.06	0.94	1	1
$\rho = 0.5$	0.08	0.98	1	1
$\rho = 0.75$	0.04	1	1	1

Quadratic effect

GRP-test Results

	$\sigma = 0$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$
$\rho = 0.25$	0.02	0.16	0.42	0.68
$\rho = 0.5$	0.02	0.28	0.48	0.6
$\rho = 0.75$	0.06	0.34	0.4	0.48

Benchmark Results

	$\sigma = 0$	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$
$\rho = 0.25$	0.06	1	1	1
$\rho = 0.5$	0.08	1	1	1
$\rho = 0.75$	0.04	1	1	1

From the above results it can be seen that, the GRP-test does indeed control the type I error, and suffers only a relatively small loss in power compared with the oracle GRP-test.

Also, empirical distribution functions of p-values for testing the goodness-of-fit in the two scenarios were drawn.

The plots with respect to each scenario is given below.

Interaction effect

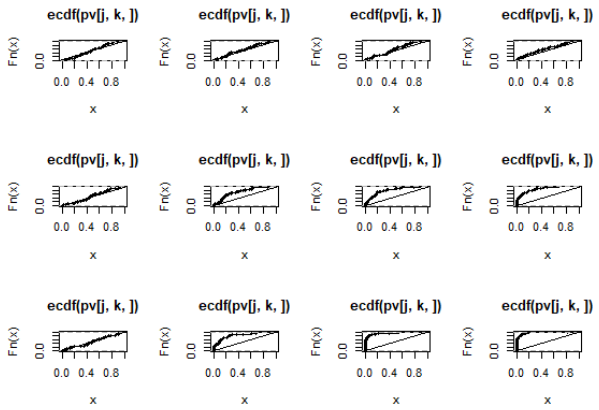

Figure 1: Empirical distribution functions of p-values for testing the goodness-of-fit with interaction effect

Quadratic effect

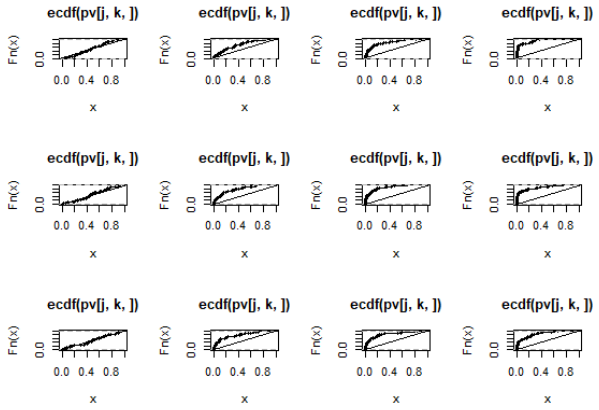

Figure 2: Empirical distribution functions of p-values for testing the goodness-of-fit with quadratic effect

4.2 Real world example

The dataset named as the "Toxicity Dataset" with observations of molecules designed for functional domains of a core clock protein, CRY1, responsible for generating circadian rhythms was used in this part.

4.2.1 Method

The dataset comprised of 1203 molecular descriptors with respect to 172 molecules. To reduce dimensionality and focus on the most informative molecular descriptors, we selected the top 500 molecular descriptors with the largest variances.

After scaling the variables, we applied logistic regression with respect to the models explained in the chapter before: lasso, ridge, adaptive lasso, and de-bieased lasso. This resulted in a parameter estimate $\hat{\beta}$ and its corresponding support set $\hat{S}$, indicating the selected variables.

Subsequently, a Gaussian model was fitted to the rows of the design matrix, capturing the dependence structure among the variables. To augment the dataset, they generated 172 additional observation vectors by simulating from this fitted model.

For generating new responses, we used a Bernoulli distribution with probabilities $\hat{\pi}(u)$. The probability of success $\hat{\pi}(u)$ is defined as a function of u and the estimated logistic regression coefficients $\hat{\beta}$. The additional term $g(u) - \bar{g}\hat{\sigma}_g 1\{\hat{\sigma}_g \neq 0\}$ introduces a non-linear component.

$g(u)$ can take three forms: 0, $u_{j_1}^2 + u_{j_2}^2$, or $u_{j_1}u_{j_2} + u_{j_3}u_{j_4}$, where $j_i \in \hat{S}$. $\bar{g}$ is the mean of the g values calculated from the original dataset. $\hat{\sigma}_g^2$ is an estimate of the variance of the g values from the original dataset.

The scenarios represent different forms of the non-linear component, introducing various patterns of dependence among the molecular descriptors. The uniformly sampled entries from $\hat{S}$ ensure that the non-linearities are introduced only in the selected variables. Finally the goodness-of-fit test was conducted for each of the varying cases and models.

4.2.2 Results

The rejection probabilities for all three scenarios from 100 repetitions were reported.

	lasso	ridge	adaptive lasso	de-biased lasso
$g(u) = 0$	0.11	0.07	0.12	0.09
$g(u) = u_{j1}^2 + u_{j2}^2$	0.39	0.46	0.36	0
$g(u) = u_{j1}u_{j2} + u_{j3}u_{j4}$	0.55	0.11	0.09	0

From the results displayed above, it can be said that, in each case, the GRP-test has been able to detect the misspecification relatively reliably, while keeping the type I error under control except for the de-biased lasso application.

5 Discussion

In conclusion, it can be stated that the GRP-test is able to detect the misspecification relatively reliably, while keeping the type I error under control. De-biased lasso method might not be effective in capturing the linear component of the model. The parameters used or the data itself might be leading to degenerate solutions, causing the algorithm to terminate without finding a non-zero solution.

6 Take-home Final Exam

6.1 Question 1

Is the paper you read good enough to be applied to the data you analyze?

6.1.1 Answer

Yes, I believe so. Given that the dataset I am using consists of data with respect to molecular descriptors and the paper has applied their novel method on genes. Genes and molecular descriptors are both entities used in the field of molecular biology and bioinformatics, although they represent different aspects of biological systems. Hence I believe the paper is good enough to be applied to the data that I analyzed.

6.2 Question 2

What is the challenge of your project–computational and theoretical?

6.2.1 Answer

Some of the computational and theoretical challenges faced in implementing the project. Implementing and running simulations with a large number of variables (p = 500) and observations (n = 800) was computationally intensive. Handling the complexity of logistic regression models with quadratic effects and interaction terms added another layer of computational challenge. Sometimes the different variations of β_0 and ρ and σ values did not work with the program. Theoretical challenges arised in understanding and correctly specifying the logistic regression models with the introduced quadratic effects and interaction terms. Additionally, addressing the impact of misspecification (controlled by the parameter σ) on the model's performance was challenging. When applied to the real data set the the ρ values were fixed for different *sigma* values. In the context of a high-dimensional setting, direct comparisons for the proposed GRP-test are challenging.

6.3 Question 3

How did you solve the challenge(s)?

6.3.1 Answer

The challenges discussed above was taken care of as follows. The simulations were run multiple times with lesser number of p and n values and then, when the results seem correct the whole set was

run. Also, since Sometimes the different variations of β_0 and ρ and σ values did not work with the program, the behaviour of these values were analyzed with different problem settings. Sometimes subsets of data were used. In order to understand the theoretical aspects additional materials were referred. With respect to the real dataset, since it was difficult to handle 1200 variables 500 variables with highest variance were considered. However, it also gave the same results. Hence, a simulated example was run to check the problem setting. To establish a theoretical benchmark, an oracle GRP-test applied to a reduced design matrix that includes only variables in the active set $S = \{1, \ldots, 5\}$, effectively reducing the problem to a low-dimensional one.

6.4 Question 4

What have you done to improve the method?

6.4.1 Answer

The paper only discusses about the implementation of logistic regression with lasso, and then fitting goodness-of-fit tests. I used other methods such as ridge, adaptive lasso, and de-biased lasso to fit the logistic regression models. Then conducted the goodness-of-fit tests to check whether the GRP tests introduced detect the misspecification reliably.

7 Appendix

```
gen.data <- function(n, p, fam, quad){
  s <- 5
  beta <- 1*c(-1,1,-1,1,-1,rep(0,p-min(p,s)))   #parameters
  x <- rho^c(0:(p-1))
  sigma0 <- toeplitz(x)

  X <- mvrnorm(n, rep(0,p), sigma0)*1 #matrix(runif(n*p),n,p)#

  if(quad == TRUE){
    f =  sig * (X[,1]^2) #*X[,3])#^2 -1)#-sigma0[1,2])#*(X[,1]^2-1) #*X[,5]#^2
  }else{
    f = sig * X[,1] * X[,2]
  }

  z = f + crossprod(t(X), beta)

  if(fam == "gaussian"){
    y = z +rnorm(n)#rlaplace(n, m=0, s=5)#1*rnorm(n)
  }else{
    pr = 1/(1 + exp(-z))
    # pr <- pnorm(z)   # misspecified link: probit
```

```
    y = rbinom(n, 1, pr)
  }

  return(list(X = X, y = y))
}

n = 800
p = 500
it10 <- 50
fam <- "binomial"

# QUADRATIC
sigs <- seq(0, 3, by = 1)
quad <- TRUE

# Interaction
sigs <- seq(0, 3, by = 1)
quad <- FALSE

pv <- array(0,dim = c(3,length(sigs),it10))
pvbench <- array(0,dim = c(3,length(sigs),it10))

rhos<-c(0.25,0.5,0.75)
irhos <- c(1,2,3)

set.seed(12)
for(j in 1:length(rhos)){
  rho <- rhos[j]
  #n <- ns[j]
  ir <- irhos[j]
  print("")
  print(paste("rho:",rhos[j]))
  for(k in 1:length(sigs)){
    sig <- sigs[k]
    print(paste("sig:",sigs[k]))
    set.seed(2)
    for(i in 1:it10){
      cat(paste(i,".."))
      gd <- gen.data(n, p, fam, quad)
      X <- gd$X
      y <- gd$y

      obj.pval <- GRPtest(X, y, fam = "binomial", nsplits = 1)
      obj.pval.bench <- GRPtest(X[, 1:5], y, fam = "binomial", nsplits = 1)

      pv[ir, k, i] <- obj.pval #test.obj$pval
      pvbench[ir, k, i] <- obj.pval.bench#test.obj.bench$pval
      cat(paste(round(obj.pval,2),"=pv"))
```

```
      cat(paste(round(obj.pval.bench,2),"=bench"))
    }
  }
}

pvbench_rho1 <- apply(pvbench[1,,], 1, function(x) mean(x < 0.05))
pvbench_rho2 <- apply(pvbench[2,,], 1, function(x) mean(x < 0.05))
pvbench_rho3 <- apply(pvbench[3,,], 1, function(x) mean(x < 0.05))

p_rho1 <- apply(pv[1,,], 1, function(x) mean(x < 0.05))
p_rho2 <- apply(pv[2,,], 1, function(x) mean(x < 0.05))
p_rho3 <- apply(pv[3,,], 1, function(x) mean(x < 0.05))

library(ranger)
library(glmnet)

gen_real_x <- function(n, p) {
  X <- rnorm(n*p); dim(X) <- c(n, p)
  rel_vars <- var_order[1:p]
  X <- X %*% sqrt_cov_x_trans[rel_vars, rel_vars]
  X <- pnorm(X)
  for (j in 1:p) {
    X[, j] <- quantile(x_orig[, rel_vars[j]], X[, j])
  }
  return(X)
}

# Load your own dataset
dataset_1 <- read.csv("C:\\Users\\sanda\\OneDrive - University of Central Florida\\UCF\\3_Fall_

y <- as.integer(dataset_1$Class == "Toxic")

df <- dataset_1[,-1204]

## select 500 variables with the highest variances
vars <- apply(df, 2, var)
max.vars500 <- df[, sort(vars, decreasing = TRUE, index.return = TRUE)$ix <= 500]
x_orig <- scale(max.vars500)

n <- nrow(x_orig)
p <- ncol(x_orig)

# Transformation to normality
cov_x_trans <- cov(qnorm(apply(x_orig, 2, rank) / (n+1)))
sqrt_cov_x_trans <- chol(cov_x_trans, pivot=TRUE)
pivot <- attr(sqrt_cov_x_trans, "pivot")
oo <- order(pivot)
```

```
sqrt_cov_x_trans <- sqrt_cov_x_trans[, oo]
var_order <- order(apply(x_orig, 2, var), decreasing=TRUE)

## generate new Gaussian data
Xn <- gen_real_x(171, 500)

lasso <- function(X, y, lambda = .1, tol=1e-6, iter=100, verbose=T) {
  w = solve(crossprod(X) + diag(lambda, ncol(X))) %*% crossprod(X,y)
  tol_curr = 1
  J = ncol(X)
  a = rep(0, J)
  c_ = rep(0, J)
  i = 1
  while (tol < tol_curr && i < iter) {
    w_old = w
    a = colSums(X^2)
    l = length(y)*lambda # for consistency with glmnet approach
    c_ = sapply(1:10, function(j) sum(X[,j]*(y - X[,-j]%*%w_old[-j])))
    w = w_old
    w[c_< l & c_ > -l] = 0
    tol_curr = crossprod(w - w_old)
    i = i + 1
    if (verbose && i%%10 == 0) message(i)
  }
  w
}
result_lasso <- lasso(x_orig, y, lambda=0.1, tol=1e-12)
result_lasso

coefs_lasso <- result_lasso
S.hat <- which(result_lasso != 0)

# calculate probabilities
pr = 1 / (1 + exp( - (Xn %*% coefs_lasso + coefs_lasso[1] + g )))

iter <- 100
pv <- rep(0, iter)

set.seed(1)
for(i in 1:iter){

  print(i)
  yn <- rbinom(nrow(Xn), 1, pr)

  test.obj <- GRPtest(Xn, yn, fam = "binomial", nsplits = 1)
  pv[i] <- test.obj
  print(test.obj)
```

```
}

mean(pv < 0.05)

###############adaptive lasso
# coordinate descent
lasso_ada <- function(X, y, adjustment, lambda = .1, tol=1e-6, iter=200, verbose=T) {
  w = solve(crossprod(X) + diag(lambda, ncol(X))) %*% crossprod(X,y)
  tol_curr = 1
  J = ncol(X)
  a = rep(0, J)
  c_ = rep(0, J)
  i = 1
  while (tol < tol_curr && i < iter) {
    w_old = w
    a = colSums(X^2)
    l = lambda/abs(adjustment) # for consistency with glmnet approach
    c_ = sapply(1:10, function(j) sum(X[,j]*(y - X[,-j]%*%w_old[-j])))
    w = w_old
    w[c_< l & c_ > -l ] = 0
    tol_curr = crossprod(w - w_old)
    i = i + 1
    if (verbose && i%%10 == 0) message(i)
  }
  w
}

#fit the adaptive Lasso estimate
result_lasso_ada <- lasso_ada(x_orig, y, result_lasso,lambda=0.5, tol=1e-12)

coefs_ada_lasso <- result_lasso_ada
S.hat <- which(result_lasso_ada != 0)

# calculate probabilities
pr = 1 / (1 + exp( - (Xn %*% coefs_ada_lasso + coefs_ada_lasso[1] + g )))

iter <- 100
pv <- rep(0, iter)

set.seed(1)
for(i in 1:iter){

  print(i)
  yn <- rbinom(nrow(Xn), 1, pr)

  test.obj <- GRPtest(Xn, yn, fam = "binomial", nsplits = 1)
  pv[i] <- test.obj
  print(test.obj)
```

```
}

mean(pv < 0.05)

# Debiased LASSO function
debiased_lasso <- function(X, y, lambda = 0.1, tol = 1e-6, iter = 100, verbose = TRUE) {
  # Fit LASSO
  lasso_coef <- lasso(X, y, lambda, tol, iter, verbose)

  # Debiased LASSO
  J <- ncol(X)
  n <- length(y)
  debias_coef <- rep(0, J)
  for (j in 1:J) {
    Xj <- X[, j, drop = FALSE]
    Sj <- setdiff(1:J, j)
    X_Sj <- X[, Sj, drop = FALSE]
    y_hat_j <- X_Sj %*% lasso_coef[Sj]
    beta_j <- lasso_coef[j] + sum(Xj * (y - y_hat_j)) / a_j
    debias_coef[j] <- beta_j
  }

  debias_coef
}

debias_lasso <- debiased_lasso(x_orig, y, lambda=0.1, tol=1e-12)

coefs_db_lasso <- debias_lasso

S.hat <- which(debias_lasso != 0)

# calculate probabilities
pr = 1 / (1 + exp( - (Xn %*% coefs_db_lasso + coefs_db_lasso[1] + g )))

iter <- 100
pv <- rep(0, iter)

set.seed(1)
for(i in 1:iter){

  print(i)
  yn <- rbinom(nrow(Xn), 1, pr)

  test.obj <- GRPtest(Xn, yn, fam = "binomial", nsplits = 1)
  pv[i] <- test.obj
  print(test.obj)
```

```
}

mean(pv < 0.05)

## We consider 3 settings as in the paper:
## g(u) = 0, g(u) = u_{j_1}^2 + u_{j_2}^2, g(u) = u_{j_1}*u_{j_2} + u_{j_3}*u_{j_4}

## NOTE: THE SETTING MUST CHOSEN MANUALLY FROM THE FOLLOWING THREE OPTIONS:

## Setting 1: g(u) = 0
g <- 0

## Setting 2: g(u) = u_{j_1}^2 + u_{j_2}^2
set.seed(1)
ids <- sample(S.hat, 2)
newvar <- Xn[,ids[1]]^2 + Xn[,ids[2]]^2
g <- sqrt(3) * scale(newvar)

## Setting 3: g(u) = u_{j_1}*u_{j_2} + u_{j_3}*u_{j_4}
set.seed(1)
ids2 <- sample(S.hat, 4)
newvar <- Xn[, ids2[1]] * Xn[, ids2[2]] + Xn[, ids2[3]] * Xn[, ids2[4]]
g <- sqrt(3) * scale(newvar)
```

References

https : *//www.stat.math.ethz.ch/ geer/Atlanta*3.*pdf*

https : *//cran.r − project.org/web/packages/hdi/hdi.pdf*

https : *//arxiv.org/pdf/*1408.4026.*pdf*

https : *//arxiv.org/pdf/*2302.14218*v*1.*pdf*

https : *//www − personal.umich.edu/ yili/DBGLMs.pdf*

https : *//online.stat.psu.edu/stat*504*/lesson/*2*/*2.4

https : *//www.investopedia.com/terms/g/goodness − of − fit*