机器学习的试验设计初探

China R Conference

郝智恒, 南开大学

什么是试验设计?

- R. A. Fisher最初在农田试验方面的工作奠定了因子设计的发展.
- 二战后，随着化工行业的发展，试验设计也迅速发展。期间提出的序贯设计思想用以控制试验数量，以控制成本。
- 随着回归模型的发展，基于模型的最优设计和响应曲面设计也逐渐发展起来。
- 超饱和设计的发展，满足了工业中用少量试验来探索从大量因子中筛选重要因子的需求。
- 计算机试验的发展，推动了空间填充设计（包括均匀设计和拉丁超立方体设计）的发展。

什么是试验设计?

如下的目的可以通过试验设计达到

- 比较试验
- 变量筛选
- 响应曲面探索
- 系统优化
- 系统稳健

More about experimental design, please refer to Wu and Hamada (2009) and Fang, Li and Sudjianto (2006).

R中比较核心的试验设计的包：

- Package AlgDesign
- Package FrF2
- Package rsm
- Package lhs

把机器学习看成做试验

- 机器学习中使用的数据，往往不是从试验中得到的。
- 但是，数据需要分为训练集和检测集。
- 我们可以利用试验设计来寻找到一个更好的训练集。如果我们设计的好，那么机器学习也会更有效或者更准确。
- 另外，在很多实际的情况下，训练数据响应值的获取是要付出高昂成本的。

接下来的部分，将使用两个基于R的模拟例子来验证上述的想法。

- 为什么用R?
- 在R里，我能找到几乎我所需要的全部。
- 非常感谢COS论坛，让我学到了很多。

Deng et al. (2009) JASA

令$X = (X_1, \ldots, X_p)^T$ 为因子向量. 响应记为Y. Y 服从二项分布. $P(Y = 1|x_0) = F(x_0)$ 是x_0 点处的$Y = 1$的概率. 我们可以定义水平为α的分类边界:

$$l_\alpha(x) = \{x : F(x) = \alpha\} \tag{1}$$

我们希望能通过训练数据来估计l_α .

为简单记, 令$p = 2$, 定义$z = wx_1 + (1 - w)x_2$, 其中w 是一个取值$[0, 1]$之间的位置的权重. 此时，有

$$F(x|\theta) = \frac{e^{(z-\mu)/\sigma}}{1 + e^{(z-u)/\sigma}}, \tag{2}$$

其中$\theta = (\mu, \sigma, w)^T$. 则我们可以推导出水平$\alpha$的分类边界为:

$$l_\alpha(x_1, x_2) = \{(x_1, x_2) : \frac{wx_1 + (1 - w)x_2 - \mu}{\sigma} = \log(\frac{\alpha}{1 - \alpha})\}. \tag{3}$$

假设训练数据记为$(x^1, Y_1), \ldots, (x^n, Y_n)$, 其中$x^i = (x_1^i, x_2^i)$. 我们给出每一个参数一个先验分布：

$$\mu \sim N(\mu_0, \sigma_\mu^2), \sigma \sim \text{Exp}(\lambda_0), w \sim \text{Beta}(\alpha_0, \beta_0). \tag{4}$$

我们便可以推导出后验分布为：

$$
\begin{aligned}
f(\theta|Y) \quad \propto \quad & \prod_{i=1}^{n} \left(\frac{e^{(wx_1^i+(1-w)x_2^i-\mu)/\sigma}}{1+e^{(wx_1^i+(1-w)x_2^i-\mu)/\sigma}} \right)^{Y_i} \\
& \left(\frac{1}{1+e^{(wx_1^i+(1-w)x_2^i-\mu)/\sigma}} \right)^{1-Y_i} \\
& \times e^{\frac{(\mu-\mu_0)^2}{-2\sigma_\mu^2}} \lambda_0 e^{-\lambda_0\sigma} w^{\alpha_0-1}(1-w)^{\beta_0-1}. \qquad (5)
\end{aligned}
$$

此时，θ 便可以通过极大似然方法来估计了

$$\hat{\theta} = \arg\max_\theta \log f(\theta|Y) \qquad (6)$$

ALSD

在迭代n次后, 我们已经估计出来了当时的分类边界如下:

$$l_{\alpha,n} = \{\hat{w}_n x_1 + (1 - \hat{w}_n)x_2 = \hat{\mu} + \hat{\sigma}\log(\frac{\alpha}{1-\alpha})\}, \qquad (7)$$

$T_{kn} = \{(x^1, Y_1), \ldots, (x^n, Y_n)\}$. 接下来，我们希望在$T_{un}$选择下一个点放入训练集. 我们从$T_{un}$ 选出k_0 个数据作为候选，这些候选数据的选取以靠近$l_{\alpha,n}$为标准.

我们将这些候选的数据记为$\tilde{x^1}, \ldots, \tilde{x^{k_0}}$. 然后按照如下方法从中选择一个数据:

$$x^{n+1} = \arg\max_{x \in \{\tilde{x^1}, \ldots, \tilde{x^{k_0}}\}} \det(I(\hat{\theta}_n, x)) \tag{8}$$

其中$I(\hat{\theta}, x)$ 是在点x处的系数的Fisher信息阵.

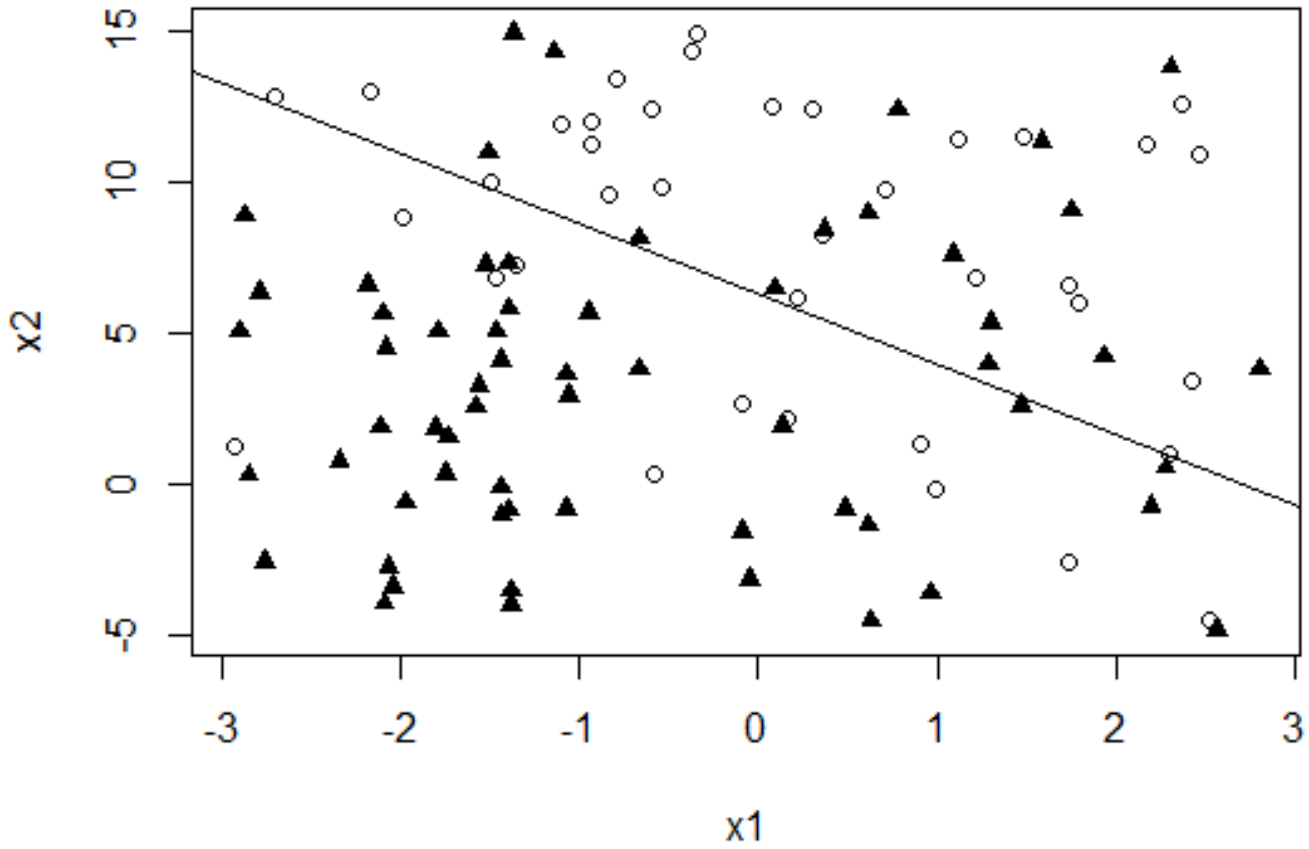
模拟例子

数据产生自分布$F(x) = \frac{exp(\frac{z-\mu}{\sigma})}{1+exp(\frac{z-\mu}{\sigma})}$ 加上服从$N(0, 0.1)$的随机误差. 产生100个随机数. 我们设定阈值$\alpha = 0.7$, 则满足$F(x) > \alpha$的x 被标记为类别1，否则被标记为类别2.

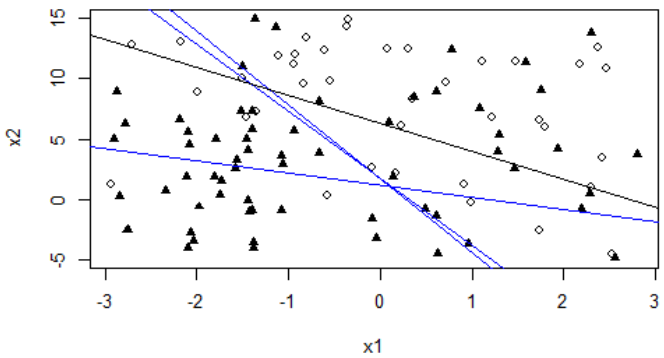
- 需要的包:numDeriv
- The complete code will not share here. If you are interested, you can contact me later.

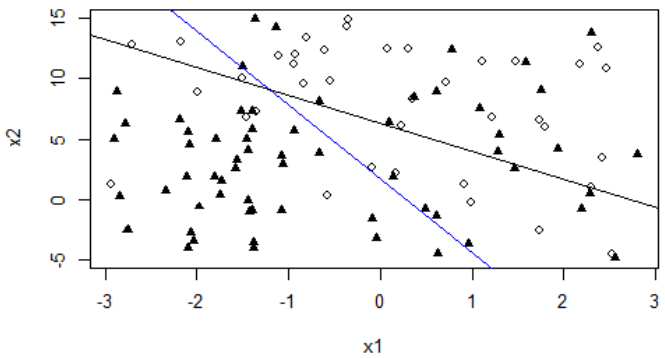
```
#开始循环
for(i in 2:50)  #i 是循环的次数
{
  data.candidate=data.frame(x1,x2)[-idx,]
  #定义一个向量，存储15个点所对应的fisher信息阵的determine
  determin=vector()
  #首先选出离分割平面最近的15个点
  ddd=distance(x=data.candidate[1],y=data.candidate[2],a=intercept[i-1],b=slope[i-1])#需
  ddd=unlist(ddd)
  r.name=row.names(data.candidate)
  dd=data.frame(ddd,r.name)
  lable=order(dd$ddd)[1:15] #距离最近的前15个点的标号
  #然后计算函数在这每一个点的fisher信息矩阵的determine
  for(j in 1:15)
  {
    ff.logit=function(parameter)
     {
       f.logit(parameter,data=data.frame(x1,x2))[lable[j],]#这里的data需要改为candidate
     }
    determin[j]=det(hessian(ff.logit,x=c(mu.hat[i-1],sigma.hat[i-1],w.hat[i-1])))
  }
  idx[i]=order(determin,decreasing=TRUE)[1]
  mle=nlminb(c(0.5,0.5,0.5),LL,data=data.frame(x1,x2,y)[idx,])
  mu.hat[i]=mle$par[1]
  sigma.hat[i]=mle$par[2]
  w.hat[i]=mle$par[3]
  intercept[i]=mu.hat[i]+sigma.hat[i]*log(4)
  slope[i]=-w.hat[i]/(1-w.hat[i])
}
```

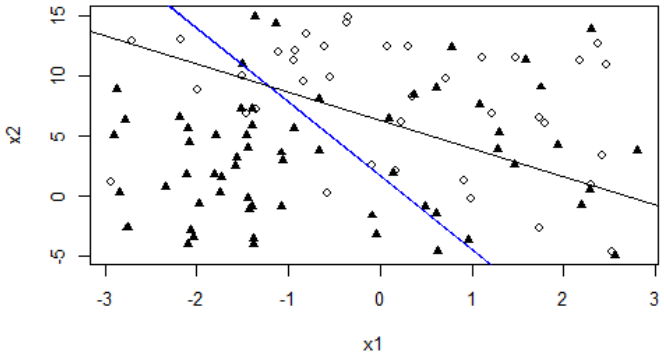
模拟结果

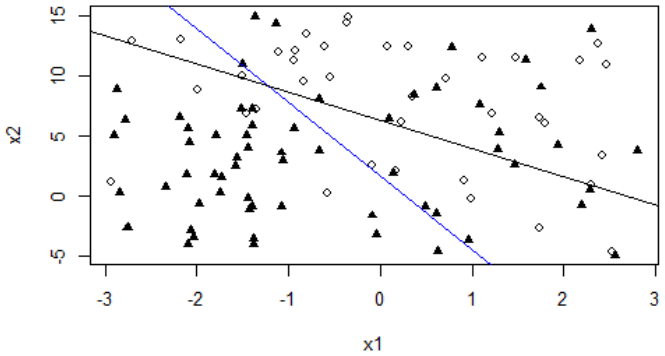


模拟结果

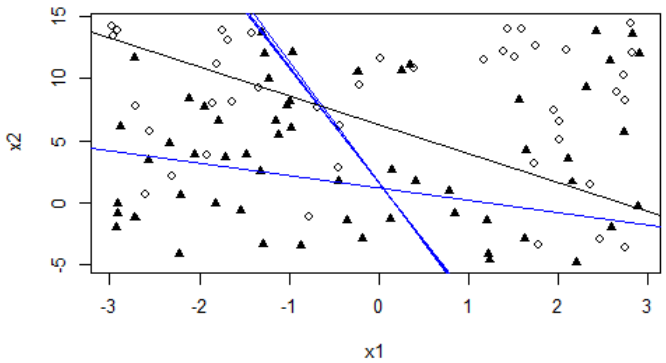

x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3

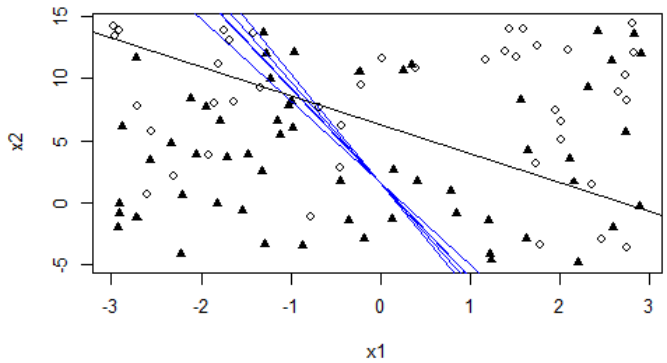

x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3

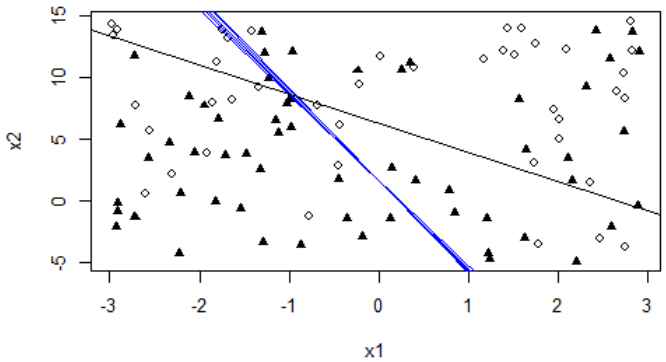

x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3

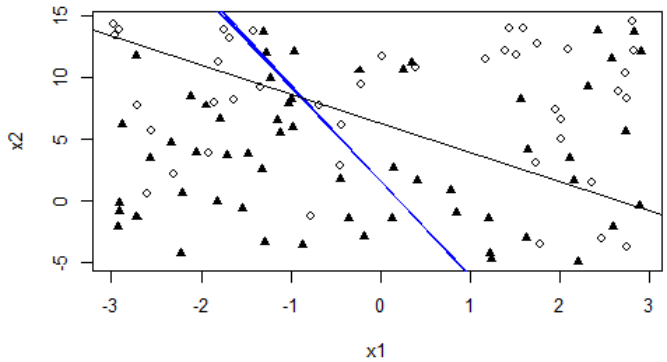

x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3

模拟结果


x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3


x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3


x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3


x2
x1
15
10
5
0
-5
-3
-2
-1
0
1
2
3

Deng, Lin and Qian (2012)

The Lasso (Tibshirani,1996) 因其收缩性，是一种非常流行的变量选择方法.
考虑如下模型

$$Y = X^T\beta + \varepsilon \tag{9}$$

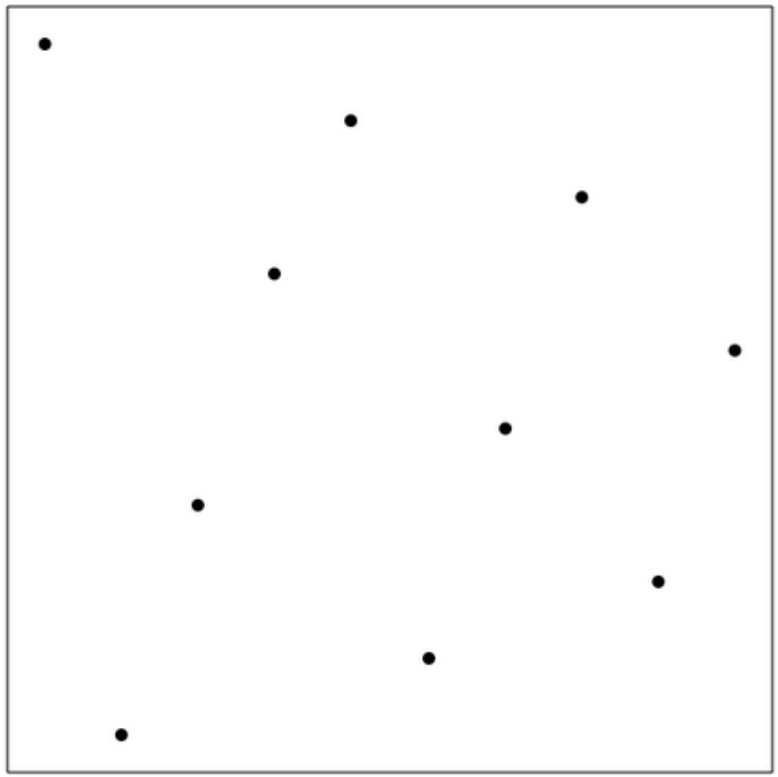
其中$X = (X_1, \ldots, X_p)^T$是一个p 维随机向量, Y 是响应, β 是回归系数, ε 服从零均值，方差为σ^2的正态分布的随机误差. 给定$n \times p$ 的设计阵X 和响应值$\mathbf{y}$, Lasso的拉格朗日解可以写成：

$$\hat{\beta} = \arg\min_\beta (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda||\beta||_{l_1} \tag{10}$$

变量选择的精确度可以用错选率来衡量, 记为ν.
令$A(\beta) = \{j : \beta_j \neq 0, j = 1, \ldots, p\}$, 则

$$p \times \nu = \sharp\{j : j \in A(\beta) \text{ but } j \bar{\in} A(\hat{\beta})\} + \sharp\{j : j \in A(\hat{\beta}) \text{ but } j \bar{\in} A(\beta)\} \tag{11}$$

之前X 都是简单随机抽样得到的. 分层抽样可以改进估计精度是众所周知的. McKay et al. (1979) 拉丁超立方体抽样(LHS), 该策略可以视作在所有变量上同时进行分层抽样.



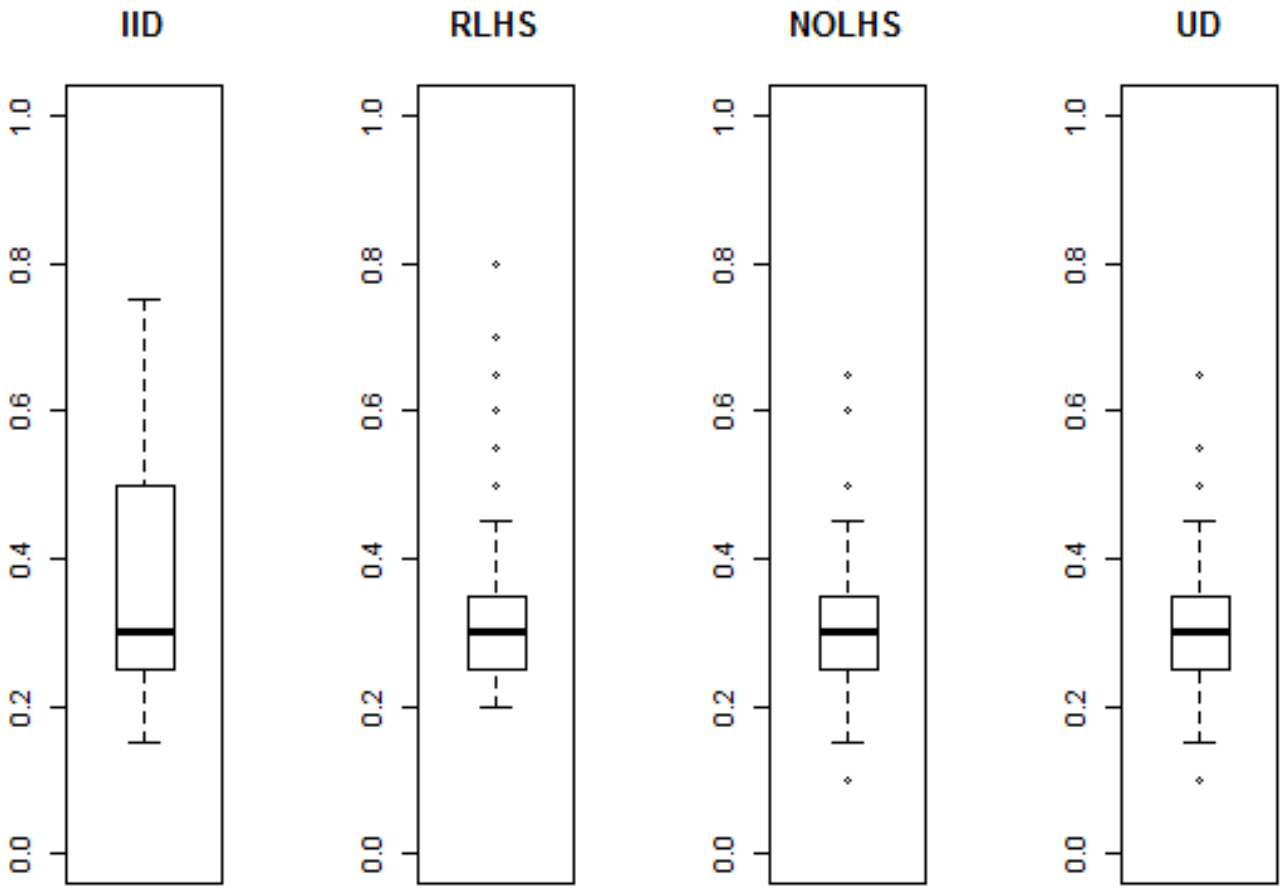
Owen (1992) showed that when using the LHS for Monte-Carlo integration, the estimator is less variate than the estimator from IID sampling. So we expect that using the LHS for Lasso can improve the solution. Also, when running a variable selection procedure, it is better the variables are less correlated so that the active variables are less correlated to the inactive variables. Orthogonal Latin hypercube design (OLHD) or nearly orthogonal latin hypercube design (NOLHD) would be a good choice. We are also motivated by Fang, Ge and Liu (2002), their result of the connection between discrepancy and $E(f_{NOD})$ in supersaturated designs maybe helpful here. Supersaturated designs are usually used for screening designs.

- 所需包: lars,lhs
- Gendex software is also needed
- Also refer to http://www.math.hkbu.edu.hk/UniformDesign/

模拟

```
response=function(x,coef)#x should be a matrix,coef should be a vector
{
  if(ncol(x)!=length(coef))
    print("error:the dimension of coefficient and of the design matrix differ")
  else
    x%*%coef+rnorm(n=nrow(x),mean=0,sd=sqrt(2))
}
#using lars for the computation of lasso solution and get the estimated coefficient
coef.estimate=function(x,y)
{
  fit.lasso=lars(x,y,type="lasso")
  cp=fit.lasso$Cp
  cp=cp[-1]
  cp.min=as.numeric(names(sort(cp)[1]))#find the minimum of cp-value
  coef.estimated=as.vector(coef(fit.lasso)[cp.min,])
  coef.estimated
}
#function for computing the sign index of the coefficient
coef.idx=function(coef)
{
  coef.idx=rep(0,length(coef))
  for(i in 1:length(coef))
  {
    if(coef[i]!=0)
      coef.idx[i]=1
  }
  coef.idx
}
```

模拟



模拟结果

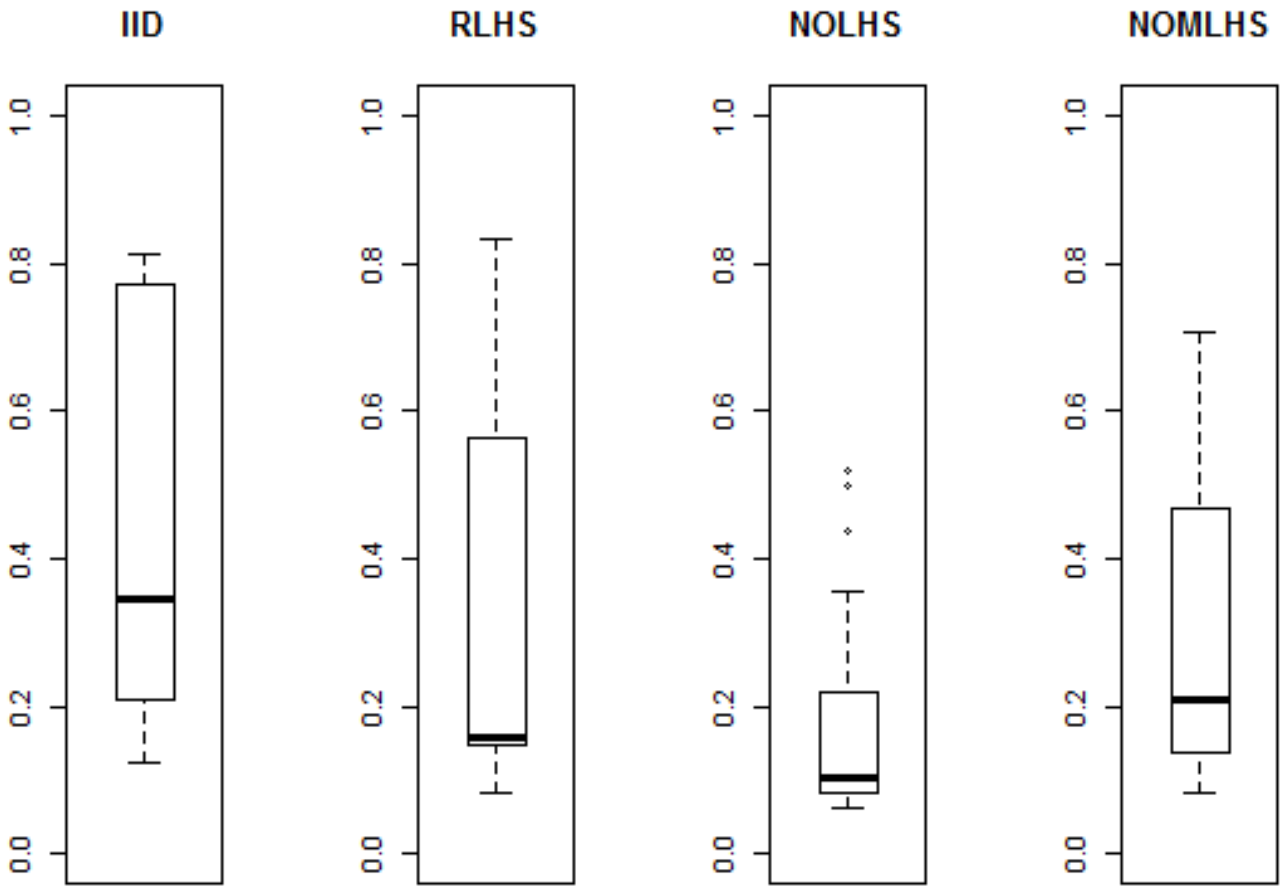


Table: The summary of ν for Case 2

	1st.Quant	Median	Mean	3rd.Quant	$\max\|\rho_{ij}\|$
IID	0.2135	0.3438	0.4638	0.7656	0.4595441
RLHD	0.1458	0.1562	0.3104	0.5052	0.4853301
NOLHD	0.0833	0.1042	0.1778	0.2031	0.01195678
NOMLHD	0.1354	0.2083	0.2083	0.4688	0.3153421

THANK YOU

ANY QUESTION???