Package ‘mdpd’

Version 1.0.0

Title minimum density power divergence

Description This package aims to estimate the coefficient of the high-dimensional linear regression model. Significantly different from the existing studies, we adopt loss functions based on MDPD (minimum density power divergence) criteria. Multiple published studies have shown that this approach is compared with alternatives in the low dimensional situation. We extend this method into high dimensional situation, and also find it is robust. Penalization is used for identification and regularized estimation. Computationally, we develop an effective algorithm which utilizes the coordinate descent. Simulation shows that the proposed approach has satisfactory performance.
License GPL-3

Author Yangguang Zang <yangguang.zang@gmail.com>; Qingzhao Zhang

<qzzhang.wise@gmail.com>; Shuangge Ma< shuangge.ma@yale.edu>

Maintainer Yangguang Zang

Date 2016-09-02

MATLAB topics documented:

❖ ar

Description

Function ar aims to generate an AR correlation matrix \Sigma which satisfies \Sigma_{jk} = rho ^|j-k|

Usage

covc = ar(p1,rho)

Input

p1: the dimension of the correlation matrix

rho: controls the degree of association

Output

covc: a correlation matrix

❖ auccalc

Description

Function auccalc aims to calculate the area under the curve (AUC).

Usage

auc = auccalc(fpr,tpr)

Input

fpr: false positive rate set

tpr: true positive rate set

Output

auc: the area under the curve (AUC)

Examples

```
% This Demo shows how to use function auccalc to compute
the area under the ROC (AUC).
fprset=[0; 0; 0.0111;0.0444;0.1037;
    0.2407;0.3778;0.4519;0.5111;0.5667;
    0.6000;0.6185; 0.6481;0.6519;0.6889;
```

```
    0.7222;0.7889;0.8296;0.9481;0.9852];
tprset=[0;0.0286;0.3429;0.6571;0.8000;
    0.8857;0.8857;0.8571;0.9143;0.9143;
    0.9143;0.9143;0.9143;0.9143;0.9143;
    0.9143;0.9714;1.0000;1.0000;1.0000];
auc = auccalc(fprset,tprset);
```

❖ fprcal

Description

Function fprcal aims to calculate the false positive rate of the estimated beta

Usage

value = fprcal(betahat,betaori)

Input

betahat: estimated beta (p dimensional vector)

betaori: original beta (p dimensional vector)

Output

value: the false positive rate

Examples

```
% This Demo shows how to use functions fprcal to compute
the false positive rate.
betaori=[1;2;3;0;0;0;0;0;0];
betahat=[0.9;2.1;3.2;0.6;0.2;0.1;0;0;0];
result=fprcal(betahat,betaori);
```

❖ lassocd

Description

Function lassocd aims to estimate the coefficient of the linear regression model.

Usage

betahat=lassocd(x,y,lam)

Input

x: covariates (n x p matrix)

y: response variable (n dimensional response vector)

lam: penalty parameter

Output

betahat: coefficient of the linear regression model

Examples

```
% This Demo shows how to use function lassocd to select the
important covariates in the high-dimensional linear
regression. Besides, in this algorithm, the loss function
is the squared loss and the penalty is lasso. We use the 5-
fold cross validation to select the best tuning parameter.
tic,
clear
n = 100;
p = 100;
%%%%%%%%%%%%%%%%%%%%%%%%%
w = randn(n,p);
covc = ar(p,0);
% covc = ar(p,0.2);
% covc = ar(p,0.8);
x = w*covc^0.5;
%%%%%%%%%%%%%%%%%%%%%%%
len = 10;
beta = zeros(p,1);
beta0 = zeros(p,1);
beta(1:len) = 0.5+rand(len,1);

ccc = 1;
index = zeros(n,1);
epsilon = zeros(n,1);
epsindex = randperm(n,floor(n*(1-ccc)));
epsilon(setdiff(1:n,epsindex)) = randn(floor(n*ccc),1);
epsilon(epsindex) = randn(floor(n*(1-ccc)),1)+5;
index(epsindex) = 1;

y = x*beta+epsilon;
```

```
nLambda = 20;
lambdaRatio  =  0.0001;
w = randn(n,p);
for i = 1:100
%     i
    lambdai = i;
    beta0 = lassocd(x,y,lambdai);
if sum(abs(beta0)<=0.001) == p
    break;
end
end
lambdaMax = i;
lambdaMin  =  lambdaMax * lambdaRatio;
loghi  =  log(lambdaMax);
loglo  =  log(lambdaMin);
lograngе  =  loghi - loglo;
interval  =  -logrange/(nLambda-1);
lambda  =  exp(loghi:interval:loglo)';
betaset = zeros(p,nLambda);
sigmaset = zeros(1,nLambda);
bicset = zeros(nLambda,1);
for i = 1:nLambda
%     i
    for ttt = 1:5
        xtest = x((ttt-1)*n/5+1:ttt*n/5,:);
        ytest = y((ttt-1)*n/5+1:ttt*n/5);
        xtrain = x(setdiff(1:n,(ttt-1)*n/5+1:ttt*n/5),:);
        ytrain = y(setdiff(1:n,(ttt-1)*n/5+1:ttt*n/5));
   [beta0,sigmaset(i)]=lassocd(xtrain,ytrain,lambda(i);
    bicset(i,1) = bicset(i,1)+sum(ytest-xtest*beta0).^2;
    end
end
for i = 1:nLambda
%     i
    [beta0,sigmaset(i)] = mdpd(x,y,lambda(i),alpha);
    betaset(:,i) = beta0(:);
end
bicbesti = find(bicset == min(bicset));
betamulti = betaset(:,bicbesti(1));
toc
```

❖ mdpd

Description

Function mdpd aims to estimate the coefficient of the linear regression model based on the MDPD method

Usage

[beta, sigma0] = mdpd(x,y,lambda,alpha)

Input

x: covariates (n x p matrix)

y: response variable (n dimensional response vector)

lambda, alpha: tuning parameters

Output

beta: estimated coefficient (p dimensional vector)

sigma0: estimation of the error variance

Examples

```
% This Demo shows how to use functions mdpd to estimate the
coefficients of the high dimensional linear model. We also
use the 5-fold cross validation method to select the best
lambda.
tic,
clear
n = 100;
p = 100;
alpha = 0.1;
%%%%%%%%%%%%%%%%%%%%%%%%%
w = randn(n,p);
covc = ar(p,0);
% covc = ar(p,0.2);
% covc = ar(p,0.8);
x = w*covc^0.5;
%%%%%%%%%%%%%%%%%%%%%%%
len = 10;
beta = zeros(p,1);
beta0 = zeros(p,1);
beta(1:len) = 0.5+rand(len,1);

ccc = 1;
index = zeros(n,1);
```

```
epsilon = zeros(n,1);
epsindex = randperm(n,floor(n*(1-ccc)));
epsilon(setdiff(1:n,epsindex)) = randn(floor(n*ccc),1);
epsilon(epsindex) = randn(floor(n*(1-ccc)),1)+5;
index(epsindex) = 1;

y = x*beta+epsilon;
nLambda = 20;
lambdaRatio  =  0.0001;
w = randn(n,p);
for i = 1:100
%     i
    lambdai = i;
    beta0 = mdpd(x,y,lambdai,alpha);
if sum(abs(beta0)<=0.001) == p
    break;
end
end
lambdaMax = i;
lambdaMin  =  lambdaMax * lambdaRatio;
loghi  =  log(lambdaMax);
loglo  =  log(lambdaMin);
logrange  =  loghi - loglo;
interval  =  -logrange/(nLambda-1);
lambda  =  exp(loghi:interval:loglo)';
betaset = zeros(p,nLambda);
sigmaset = zeros(1,nLambda);
bicset = zeros(nLambda,1);
for i = 1:nLambda
%     i
    for ttt = 1:5
        xtest = x((ttt-1)*n/5+1:ttt*n/5,:);
        ytest = y((ttt-1)*n/5+1:ttt*n/5);
        xtrain = x(setdiff(1:n,(ttt-1)*n/5+1:ttt*n/5),:);
        ytrain = y(setdiff(1:n,(ttt-1)*n/5+1:ttt*n/5));
    [beta0,sigmaset(i)] =
mdpd(xtrain,ytrain,lambda(i),alpha);
    bicset(i,1) = bicset(i,1)+sum(ytest-xtest*beta0).^2;
    end
end
for i = 1:nLambda
%     i
    [beta0,sigmaset(i)] = mdpd(x,y,lambda(i),alpha);
    betaset(:,i) = beta0(:);
end
bicbesti = find(bicset == min(bicset));
betamulti = betaset(:,bicbesti(1));
```

```
toc
```

❖ tprcal

Description

Function tprcal aims to calculate the true positive rate of the estimated beta

Usage

value = tprcal(betahat,betaori)

Input

betahat: estimated beta (p dimensional vector)

betaori: original beta (p dimensional vector)

Output

value: the true positive rate

Examples

```
% This Demo shows how to use functions tprcal to compute
the true positive rate.
betaori=[1;2;3;0;0;0;0;0;0];
betahat=[0.9;2.1;3.2;0.6;0.2;0.1;0;0;0];
result=tprcal(betahat,betaori);
```

References

1. Basu, A., Harris, I. R., Hjort, N. L., and Jones, M. (1998). Robust and efficient estimation by minimising a density power divergence. Biometrika, 85(3), 549–559.
2. Durio, Alessandra and Isaia, Ennio Davide (2011). The minimum density power divergence approach in building robust regression models. Informatica, 22(1), 43– 56.
3. Ghosh, Abhik and Basu, Ayanendranath and others (2013). Robust estimation for independent non-homogeneous observations using density power divergence with applications to linear regression. Electronic Journal of statistics, 7, 2420–2456.
4. Wu, Tong Tong, and Kenneth Lange. "Coordinate descent algorithms for lasso penalized regression." *The Annals of Applied Statistics* (2008): 224-244.