Package ‘gre’

Version 1.0.0

Title High-dimensional general relative error

Description This package aims to estimate the joint effects of multiple main effects and interactions in the high-dimensional linear multiplicative (or accelerated failure time - AFT) model. It utilizes the general relative error (GRE) criteria to build the objective function. Significantly different from the existing studies, we adopt loss functions based on the relative errors, which offer a useful alternative to the “classic” methods such as the least squares and least absolute deviation. Further to accommodate censoring in the response variable, we adopt a weighted approach. Penalization is used for identification and regularized estimation. Computationally, we develop an effective algorithm which combines the majorize-minimization and 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@gmail.com>

Maintainer Yangguang Zang

Date 2016-05-16

MATLAB topics documented:

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 functions 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);
```

❖ km

Description

Function km creates the Kaplan-Meier weight for right censoring data.

Usage

weight=km(y,Delta)

Input

y: n dimensional right censoring observation.

Delta: n dimensional event indicator, which has the same length with y. The element of Delta equals 1 if the event was observed, or equals 0 if the responce was censored

Output

weight: n dimensional Kaplan-Meier vector.

Examples

```
% This Demo shows how to use function km to estimate the
weight when the data is right censoring.
clear
n = 200;
p1 = 500;
p2 = 5;
```

```
w = randn(n,p1);
x1 = w;
x2 = randn(n,p2);
x3 = zeros(n,p1*p2);
for i = 1:n
x3(i,:) = reshape(x2(i,:)'*x1(i,:),p1*p2,1);
end
x = [x1,x2,x3];
[~,p] = size(x);
beta01 = zeros(p1,1);
beta02 = zeros(p2,1);
beta03 = zeros(p1*p2,1);
beta01(1:10) = 0.4+0.8*rand(10,1);
beta02(1:5) = 4+0.8*rand(5,1);
beta03(1:20) = 4+0.8*rand(20,1);
beta = [beta01;beta02;beta03];
YY = exp(x*beta).*exp(randn(n,1));
C = 100*rand(n,1);
y = zeros(n,1);
Delta = zeros(n,1);
for i = 1:n
y(i) = min(C(i),YY(i));
if YY(i) <= C(i)
Delta(i) = 1;
end
end
weight = km(y,Delta);
```

❖ lare

Description

Function lare estimates the coefficients of the high dimensional AFT model with the least add relative errors (LARE) method. We use the MCP penalty in this function.

Usage

betahat = lare(x,y,lam,rr,weight)

Input

x: n x p covariates

y: n dimensional response vector

lam, rr: tuning parameters

weight: the weight allocated to the samples (n dimensional vector).

Default as 1/n. n is the number of samples.

Output

betahat: estimated coefficient (p dimensional vector)

Examples

```
% This Demo shows how to use functions lare to estimate the
coefficients of the high dimensional AFT model. We also use
the 5-fold cross validation method to select the best
lambda.
tic,
clear
n = 100;
p = 200;
maxit = 20;
maxit1 = 20;
toler = 1E-4;
nLambda = 20;
rr = 6;
lambdaRatio = 1E-5;
w = randn(n,p);
wtest = randn(n,p);
x = w;
beta = zeros(p,1);
beta(1:10) = 0.4+0.8*rand(10,1);
y = exp(x*beta).*exp(randn(n,1));
beta0 = zeros(p,1);
for i = 1:100
lam = i;
betahat = lare(x,y,lam,6);
if sum(abs(betahat) <= 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);
bicset = zeros(nLambda,1);
for i = 1:nLambda
i
for ttt = 1:5
xtest = x(floor((ttt-1)*n/5)+1:floor(ttt*n/5),:);
ytest = y(floor((ttt-1)*n/5)+1:floor(ttt*n/5),:);
xtrain = x(setdiff(1:n,floor((ttt-
1)*n/5)+1:floor(ttt*n/5)),:);
ytrain = y(setdiff(1:n,floor((ttt-
1)*n/5)+1:floor(ttt*n/5)),:);
beta0 = lare(xtrain,ytrain,lambda(i),6);
bicset(i,1) = bicset(i,1)+sum((log(ytest)-xtest*beta0).^2);
end
end
bicbesti = find(bicset == min(bicset(bicset>0)));
for i = bicbesti
betahat = lare(x,y,lambda(i),6);
betaset(:,i) = betahat;
end
finalbeta = betaset(:,bicbesti);
plot(bicset)
toc
```

❖ lpre

Description

Function lpre estimates the coefficients of the high dimensional AFT model with the least product relative errors (LPRE) method. We use the MCP penalty in this function.

Usage

betahat = lpre(x,y,lam,rr,weight)

Input

x: n x p covariates

y: n dimensional response vector

lam, rr: tuning parameters

weight: the weight allocated to the samples (n dimensional vector).

Default as 1/n. n is the number of samples.

Output

betahat: estimated coefficient (p dimensional vector)

Examples

```
% This Demo shows how to use functions lpre to select the
important genes, environment factors, and gene-environment
interactions under the AFT model. Note that in the Demo the
response is also right censoring, so we weight samples with
the Kaplan-Meier weight. The best lambda is chosen as
follows: we generate a same structure data without
censoring as the test set, and select the tuning parameter
which makes the sum of squares errors of the test set
smallest. We also can use the 5-fold cross validation to
select the best tuning parameter.
tic,
clear
n = 200;
p1 = 50;
p2 = 5;
maxit = 20;
maxit1 = 20;
toler = 1E-4;
nLambda = 20;
rr = 6;
lambdaRatio = 1E-5;
w = randn(n,p1);
wtest = randn(n,p1);
x1 = w;
x2 = randn(n,p2);
x3 = zeros(n,p1*p2);
x1test = wtest;
x2test = randn(n,p2);
x3test = zeros(n,p1*p2);
for i = 1:n
x3(i,:) = reshape(x2(i,:)'*x1(i,:),p1*p2,1);
end
for i = 1:n
x3test(i,:) = reshape(x2(i,:)'*x1(i,:),p1*p2,1);
end
x = [x1,x2,x3];
xtest = [x1test,x2test,x3test];
[~,p] = size(x);
```

```
beta01 = zeros(p1,1);
beta02 = zeros(p2,1);
beta03 = zeros(p1*p2,1);
beta01(1:10) = 0.4+0.8*rand(10,1);
beta02(1:5) = 0.4+0.8*rand(5,1);
beta03(1:20) = 0.4+0.8*rand(20,1);
beta = [beta01;beta02;beta03];
YY = exp(x*beta).*exp(randn(n,1));
YYtest = exp(xtest*beta).*exp(randn(n,1));
% YY = exp(x*beta).*exp(4*rand(n,1)-2);
% epsilon = zeros(n,1);
% for i = 1:n
% cc = rand();
% if cc<0.9
% epsilon(i) = randn();
% else
% epsilon(i) = trnd(1);
% end
% end
% YY = exp(x*beta).*exp(epsilon);
C = 100.*rand(n,1);
y = zeros(n,1);
Delta = zeros(n,1);
for i = 1:n
y(i) = min(C(i),YY(i));
if YY(i) <= C(i)
Delta(i) = 1;
end
end
yold = y;
[y,index] = sort(y);
x = x(index,:);
Delta = Delta(index,:);
weight = km(y,Delta);
r = 6;
beta0 = zeros(p,1);
for i = 1:100
lam = i;
betahat = lpre(x,y,lam,6);
if sum(abs(betahat) <= 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);
tprset = zeros(nLambda,1);
fprset = zeros(nLambda,1);
ebicset = zeros(nLambda,1);
for i = 1:nLambda
beta0 = lpre(x,y,lambda(i),6);
betaset(:,i) = beta0;
tprset(i,1) = tprcal(beta0,beta);
fprset(i,1) = fprcal(beta0,beta);
ebicset(i,1) = 2*log(sum((log(YYtest)-xtest*beta0).^2));
end
auc = auccalc(fprset,tprset);
ebicbesti = find(ebicset == min(ebicset(ebicset>0)));
finalbeta = betaset(:,ebicbesti);
plot(ebicset)
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);
```

Reference

1. Chen, K., Guo, S., Lin, Y., & Ying, Z. (2012). Least absolute relative error estimation. *Journal of the American Statistical Association*.
2. Chen, Kani, et al. "Least product relative error estimation." *Journal of Multivariate Analysis* 144 (2016): 91-98.
3. Hunter, David R., and Runze Li. "Variable selection using MM algorithms." *Annals of statistics* 33.4 (2005): 1617.
4. Wu, Tong Tong, and Kenneth Lange. "Coordinate descent algorithms for lasso penalized regression." *The Annals of Applied Statistics* (2008): 224-244.
5. Zang, Y.G., Zhao, Y.J., Zhang, Q.Z., Chai, H., Zhang, S.G., and Ma, S. (2016). “Identifying Gene-Environment Interactions with a Least Relative Error Approach”. *arXiv preprint arXiv:1605.09000.*