Package 'spa'

February 20, 2015

Type Package

Title Implements The Sequential Predictions Algorithm

Version 2.0

Date 2010-7-22

Author Mark Culp

Maintainer Mark Culp <mculp@stat.wvu.edu>

Depends R (>= 2.10), cluster, MASS

Description Implements the Sequential Predictions Algorithm

License GPL

Repository CRAN

Date/Publication 2012-07-23 10:35:33

NeedsCompilation no

R topics documented:

coraAI *The Cora AI Data*

Description

The coraAI data consists of a response, journal indication matrix, and co-citation network. This data is a subset of the Cora text mining project (refer to reference).

The observations are text documents that consist of 879 published papers about either Artificial Intelligence (AI) or Machine Learning (ML). The journal name for each document is available (8 journals and an other category). The observed co-citation graph is also available, where each vertex is a document (observation), and the edge is the count of citations in common between each document and all other documents.

The goal is to incorporate both the text information and co-citation information for the prediction of paper subject AI/ML. Another, interesting problem might be to predict the journal of the paper given the text information and the categorization.

Usage

```
data(coraAI)
```

Format

The coraAI data consists of three objects each discussed next.

class: categorization of the document(observation) as either AI or ML. Typically the response.

journals: indication of the document as published in a specific journal, (other, artificial-intelligence, machine-learning, nueral-computing, ieee-trans-Nnet, ieee-tpami, j-artificial-intelligence-research, ai-magazine, JASA)

cite: the adjacency matrix of the co-citation network for these 879 documents.

Details

The spa is particularly appealing for this data since it fits a function directly to the graph and coeficient vector to the journals. Other approaches require convergence of the journal information into a graph for processing, which is unclear when the data is a binary design matrix.

Source

The data was generated using AWK scripting from the cora raw sweet (first reference). The journal names were fixed to obtain a useable representation (e.g. tpami, ieee tpami, pami are all ieee-tpami).

References

A. McCallum, K. Nigam, J. Rennie, and K. Seymore (2000). Automating the construction of internet portals with machine learning. Information Retrieval Journal, 3.

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

floyd *Shortest Path Distances*

Description

'uDist' Computing shortest path distance (hop count) for an unweighted similarity graph where 1=edge and 0=no edge (i.e. invoke knnGraph with weight=FALSE)

'floyd' Floyd's algorithm (SLOW) for computing shortest path distances (weighted hop count) for distance weighted graphs (i.e. invoke knnGraph with weight=TRUE).

Usage

```
floyd(x,verbose=FALSE)

uDist(g, k)
```

Arguments

x	the adjacency n by n matrix (distance weighted)
verbose	print the progress for Floyd's algorithm
g	the adjacency n by n matrix (unweighted)
k	input parameter for determining the maximum distance to compute (a distance of greater than k is taken as infinity)

Details

These implementations are not yet optimized for large data sets (>5000 observations)

Value

D	Distance Matrix n by n

Author(s)

Mark Culp

See Also

spa knnGraph epsGraph

knnGraph *Convert a data matrix to a K-NN graph or epsilon graph*

Description

These functions are designed to convert data sets or distance matrices to graphs.

Usage

```
knnGraph(x, k = 5, weighted = TRUE, dist = FALSE, ...)
epsGraph(x, eps = 0.2, weighted = TRUE, dist=FALSE, ...)
```

Arguments

x	the data matrix, if dist=TRUE then x is a distance matrix
k	the number of neighbors for a knn graph
eps	size of fixed radius
weighted	if TRUE the graph will be weighted. Note if TRUE use floyd for shortest path distance otherwise use fastDist.
dist	when dist=TRUE then x is a distance matrix
...	Extra parameters for the daisy(...) method

Details

The daisy function in cluster library is used primarily due to its speed and reliability.

Value

g	A graph made from data x or distance matrix as input.

Note

This command relies on the cluster library

Author(s)

Mark Culp

References

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

See Also

spa knnGraph epsGraph

plot.spa *Plot the graph*

Description

Plots the graph in two specified dimensions.

Usage

```
## S3 method for class 'spa'
plot(x,xscale=NULL,g=NULL,vars=1:2,
                  graph.grid=TRUE,plt.response,
                  lcol=8,llwd=1,...)
```

Arguments

x	the spa object.
xscale	the dimension space for plotting. See spa for examples with ISOmap and data.
g	graph adjacency matrix.
vars	variables to read off of xscale, e.g. xscale[,var] (default=1:2)
graph.grid	whether or not the graph edges should be plotted.
plt.response	whether or not the graph edges should be plotted.
lcol	line color for graph edges(default=gray)
llwd	line thickness for graph edges (default=2)
...	additional parameters for plot

Author(s)

Mark Culp

References

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

predict.spa *Inductive Predict Procedure For SPA*

Description

This implements the inductive prediction for an spa object. The impact of the observation(s) on the existing labeled and unlabeled data is ignored.

For transductive predicton use update function.

Usage

```
## S3 method for class 'spa'
predict(object,xnew,gnew,type=c("vector","probs","both"),...)
```

Arguments

object	an existing object of type spa
xnew	the new xdata to predict, of diminsion ngXdim(object[2]) (only applicable if object was called with x data)
gnew	the new graph links. The diminsion on this matrix is ngX dim(object)[1] with ng as the number of observations to predict.
type	the type of prediction to return.
...	additional arguments passed into the function.

Author(s)

Mark Culp

References

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

Examples

```
## Use simulated example (two moon) and generate a contour plot of the border
set.seed(100)
dat=spa.sim(type="moon")

L=which(!is.na(dat$y))
U=which(is.na(dat$y))
Dij=as.matrix(daisy(dat[,-1]))

##Use spa to train with a supervised/transductive kernel smoother
gsup<-spa(dat$y[L],graph=Dij[L,L],control=spa.control(gcv="lGCV"))
gsemi<-spa(dat$y,graph=Dij,control=spa.control(gcv="aGCV"))
```

```
##Use predict to define a grid for contour plot
gsize=100
a1<-seq(min(dat$x1),max(dat$x1),length=gsize)
a2<-seq(min(dat$x2),max(dat$x2),length=gsize)

zsup=matrix(0,gsize,gsize)
for(i in 1:gsize){
  val=sqrt(t(sapply(1:gsize,function(j)(dat$x1[L]-a1[i])^2+(dat$x2[L]-a2[j])^2)))
  zsup[i,]=predict(gsup,gnew=val)  ##supervised prediction with k-smoother
}
zsemi=matrix(0,gsize,gsize)
for(i in 1:gsize){
  val=sqrt(t(sapply(1:gsize,function(j)(dat$x1-a1[i])^2+(dat$x2-a2[j])^2)))
  zsemi[i,]=predict(gsemi,gnew=val) ##inductive prediction with transductive rule
}

## Plot results
gr=c(2,4)
plot(0,1,cex=1.5,xlab="x1",ylab="x2",type="n",
     xlim=c(-0.5,6.5),ylim=c(-6.5,3.5))
gr=gray(c(.7,.9))
points(dat$x1,dat$x2,pch=16,col=8,cex=1)
points(dat$x1[L],dat$x2[L],pch=16,col=gr[dat$y[L]+1],cex=2)
points(dat$x1[L],dat$x2[L],pch=1,col=1,cex=2)
contour(a1,a2,zsup, levels=.5,add=TRUE,lty=1,method="edge",drawlabels=FALSE,lwd=2,col=gray(0.6))
contour(a1,a2,zsemi,levels=.5,add=TRUE,lty=1,method="edge",drawlabels=FALSE,lwd=2,col=1)
exp1=expression(hat(Y)*"="*hat(f)(G[X]))
exp2=expression(Y[L]*"="*f(X[L])+epsilon)
legend(4.25,par()$usr[4],c(exp1,exp2),cex=1.24,fill=gray(c(0,0.6)),bg="white")
legend(4.25,par()$usr[4],c(exp1,exp2),cex=1.24,fill=gray(c(0,0.6)))

title(switch(attr(dat,"type"),moon="Two Moon Simulated Data",supervised="Supervised Border Simulated set"))
title("\n\nInductive Prediction with a Transductive Model",cex.main=1.1)
box()
```

---

spa *sequential prediction algorithm*

Description

This performs the sequential predictions algorithm 'spa' in R as described in the references below. It can fit a graph only estimate (y=f(G)) and graph-based semi-parametric estimate (y=Xb+f(G)). Refer to the example below.

The approach distinguishes between inductive prediction (`predict` function) and transductive prediction (`update` function). This is documented in the user manual and references.

Usage

```
spa(y,x,graph,type=c("soft","hard"),kernel=function(r,lam=0.1){exp(-r/lam)},
    global,control,...)
```

Arguments

y	response of length m<=n
x	n by p predictor data set (assumed XU is in space spanned by XL).
graph	n by n dissimilarity matrix for a graph.
type	whether soft labels (squared error), or hard labels (exponential). Soft is default.
kernel	kernel function (default=heat)
global	(optional) the global estimate to lend weight to (default is mean of known responses).
control	spa control parameters (refer to spa.control for more information)
...	Currently ignored

Details

If the response is continuous the algorithm only uses soft labels (hard labels is not appropriate or sensible).

In classification the algorithm distinguishes between hard and soft labeled versions. To use hard labels both type="hard" must be set and the response must be two leveled (note it does not have to be a factor, also classification of a set-aside x data is not possible). The main issue between these involves rounding the PCE at each iteration (hard=yes, soft=no). If soft labels are used then the base algorithm converges to a closed form solution, which results in fast approximations for GCV, and direct implementation of that solution as opposed to iteration (currently implemented). For hard labels this is not the case. As a result approximate GCV and full GCV are not properly developed and if specified the procedure performs them with the soft version for parameter estimation.

The update function also employs a distinction between hard/soft labels. For hard labels the algorithm employs the pen=hlasso (hyperbolic l1 penalty) whereas soft labels employs the pen=ridge. One can also use the ridge penalty with hard labels but it is uncertain why this would be considered.

The code provides semi-supervised graph-based support for R.

Note

To control parameter estimation, the parameters lmin, lmax and ldepth are set through spa.control. For this procedure GCV is used as the criteria, where unlabeled data influence GCV. Use spa.control to set this as well. Options include, agcv for approximate transductive gcv, fgcv for gcv applied to the full smoother, lgcv for labeled data only or supervised gcv, and tgcv for pure transductive gcv (slow). The fgcv flag has been depreciated. Refer to spa.control and the references below for more.

References

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

M. Culp and G. Michailidis (2008) Graph-based Semi-supervised Learning. IEEE Pattern Analysis And Machine Intelligence. 30:(1)

Examples

```
## SPA in Multi-view Learing -- (Y,X,G) case.
## (refer to coraAI help page for more information).
## 1) fit model Y=f(G)+epsilon
## 2) fit model Y=XB+f(G)+epsilon

data(coraAI)
y=coraAI$class
x=coraAI$journals
g=coraAI$cite

##remove papers that are not cited
keep<-which(as.vector(apply(g,1,sum)>1))
y<-y[keep]
x<-x[keep,]
g=g[keep,keep]

##set up testing/training data (3.5% stratified for training)
set.seed(100)
n<-dim(x)[1]
Ns<-as.vector(apply(x,2,sum))
Ls<-sapply(1:length(Ns),function(i)sample(which(x[,i]==1),ceiling(0.035*Ns[i])))
L=NULL
for(i in 1:length(Ns)) L=c(L,Ls[[i]])
U<-setdiff(1:n,L)
ord<-c(L,U)
m=length(L)
y1<-y
y1[U]<-NA

##Fit model on G
A1=as.matrix(g)
gc=spa(y1,graph=A1,control=spa.control(dissimilar=FALSE))
gc

##Compute error rate for G only
tab=table(fitted(gc)[U]>0.5,y[U])
1-sum(diag(tab))/sum(tab)

##Note problem
sum(apply(A1[U,L],1,sum)==0)/(n-m)*100  ##Answer:  39.79849

##39.8% of unlabeled observations have no connection to a labeled one.

##Use Transuductive prediction with SPA to fix this with parameters k,l
pred=update(gc,ynew=y1,gnew=A1,dat=list(k=length(U),l=Inf))
tab=table(pred[U]>0.5,y[U])
1-sum(diag(tab))/sum(tab)

##Replace earlier gj with the more predictive transductive model
gc=update(gc,ynew=y1,gnew=A1,dat=list(k=length(U),l=Inf),trans.update=TRUE)
gc
```

```
## (Y,X,G) case to fit Y=Xb+f(G)+e
gjc<-spa(y1,x,g,control=spa.control(diss=FALSE))
gjc

##Apply SPA as transductively to fix above problem
gjc1=update(gjc,ynew=y1,xnew=x,gnew=A1,dat=list(k=length(U),l=Inf),trans.update=TRUE)
gjc1

##Notice that the unlabeled transductive adjustment provided new estimates
sum((coef(gjc)-coef(gjc1))^2)

##Check testing performance to determine the best model settings
tab=table((fitted(gjc)>0.5)[U],y[U])
1-sum(diag(tab))/sum(tab)

tab=table((fitted(gjc1)>0.5)[U],y[U])
1-sum(diag(tab))/sum(tab)
```

`spa.control` *Control Parameters for spa*

Description

Controls various aspects of fitting the 'spa' object.

Usage

```
spa.control(eps=1e-6,maxiter=20,gcv=c("lGCV","tGCV","fGCV","aGCV"),
            lqmax=0.2,lqmin=0.05,ldepth=10,ltmin=0.05,lgrid=NULL,
            lval=NULL,dissimilar=TRUE,pce=FALSE,adjust=0,warn=FALSE,...)
```

Arguments

`eps`	the tolerance parameter for spa using a type='class' argument.
`maxiter`	the maximum number of iterations to run the algorithm using type='class' argument. This parameter forces the algorithm to stop even if eps is not met.
`gcv`	aGCV=approximate GCV using the smoother SLL+t(SU)*SUL, tGCV=GCV using the smoother SLL+SLUsolve(I-SUU,SUL) (can be slow), lGCV=GCV using the supervised smoother (fast but not that good), and fGCV=approximate GCV using the smoother S with approximation above (this is no longer documented but it is still implemented).
`lqmax`	max quantile on the density of distance for data-driven estimation
`lqmin`	min quantile on the density of distance for data-driven estimation
`ldepth`	the depth of the search for divide and conquer parameter estimation
`ltmin`	the minimum value, in-case `lqmin` is negative

lgrid	if set to an integer, then the divide and conquer approach is bypassed
lval	if set then the smoothing parameter is lval
dissimilar	if the edges represent similarity then set this to TRUE. This flag is intended for use with the Laplacain smoother as input (for SPS this flag is ignored and the graph is assumed to be dissimilar). If the flag is FALSE then the supplied kernel is used to convert the graph to similarity.
warn	if TRUE then the procedure warns the user that a ginv will be used in the matrix inversion (i.e. the matrix is not invertible)
pce	parameter adjust is meant for adjusting hard probability class estimates to soft (i.e. if p(z)=1 then p(z)=0.9999), for GCV estimation, this pushes GCV away from selecting smaller values.
adjust	apply adjustment W=W+adjust.
...	mop up additional parameters passed in.

Note

Keep in mind, that for exponential loss (hard) we are being somewhat non-conventional by using GCV at all, i.e. loss/df where df=1-tr/m (m is known data size).

Author(s)

Mark Culp

References

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

---

spa.sim *Simulated data for SPA*

Description

An easy function to access the "moon" data set used to illustrate this package.

Usage

```
spa.sim(n=206,m=3,p=8,type=c("moon","supervised"),nonoise=TRUE)
```

Arguments

n	the total number of x observations to generate
m	the number of labeled cases (m<n) to generate
p	the diminsion on the x data. First two columns are meaningful, others are noise.
type	the unlabeled border to generate
nonoise	whether or not the addiotional noisy columns should be returned
...	additional arguments passed into the function

Author(s)

Mark Culp

References

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

Examples

```
set.seed(100)
dat=spa.sim(type="moon")

L=which(!is.na(dat$y))
U=which(is.na(dat$y))

##Fit data
gsemi<-spa(dat$y,graph=as.matrix(daisy(dat[,-1])))
gsemi
```

`update.spa` *Update procedure for transductive prediction with the SPA*

Description

This implements the transductive prediction for an spa object. It performs regularization/region approach for transductive prediction. In addition it can also updates an existing spa object with new transductive estimate.

Usage

```
## S3 method for class 'spa'
update(object,ynew,xnew,gnew,
                    type=c("vector","probs","coef","all"),
                    reg=c("ridge","hlasso"),trans.update=FALSE,
                    dat=list(k=0,l=Inf),verbose=FALSE,FUN=sum,...)
```

Arguments

`object`	an object of type spa
`ynew`	an object of type spa
`xnew`	an object of type spa
`gnew`	an object of type spa
`type`	the type of predictions in classification, classes, probabilities or both. In the case of both the object will return an additional penalty vector corresponding to the rate function for each case.

reg	for regression it is automatically taken as a ridge penalty. In the case of classification one can use either ridge or the hyperbolic l1 penalty (hlasso).
trans.update	comming soon
verbose	comming soon
dat	data driven estimation routine consists of list k for the number of vertex sets, and l for the regularization (see reference). default dat=list(k=0,l=Inf)
FUN	measure used to sort WUL, the unlabeled-labeled partition. The FUN=sum multiplies WUL times a vector of ones, others may include max.
...	additional arguments passed into the function

Author(s)

Mark Culp

References

M. Culp (2011). spa: A Semi-Supervised R Package for Semi-Parametric Graph-Based Estimation. Journal of Statistical Software, 40(10), 1-29. URL http://www.jstatsoft.org/v40/i10/.

Examples

```
## Use simulated example (two moon)
set.seed(100)
dat=spa.sim(type="moon")

##Use spa to train with a supervised/transductive kernel smoother
gsemi<-spa(dat$y,graph=as.matrix(daisy(dat[,-1])))

##Use spa to update the object with new data
dat<-rbind(dat,spa.sim(100,0))
gsemi<-update(gsemi,ynew=dat$y,,as.matrix(daisy(dat[,-1])),trans.update=TRUE)
```

Index