Package: supercompress (via r-universe)

October 28, 2024

Type Package

Title Supervised Compression of Big Data

Version 1.1

Imports FNN, stats

Description A supervised compression method that incorporates the response for reducing big data to a carefully selected subset. Please see Joseph and Mak (2021) <doi:10.1002/sam.11508>. This research is supported by a U.S. National Science Foundation (NSF) grant CMMI-1921646.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Author Chaofan Huang [aut, cre], V. Roshan Joseph [aut]

Maintainer Chaofan Huang <chaofan.huang@gatech.edu>

Repository CRAN

Date/Publication 2022-01-14 04:40:02 UTC

Contents

`supercompress` *Supervised Data Compression via Clustering.*

Description

`supercompress` is the supervised data compression method proposed in Joseph and Mak (2021). It is a nonparametric compression method that incorporates information of the response.

Usage

```
supercompress(n, x, y, lam = 0, standardize = TRUE)
```

Arguments

n	number of compressed data points
x	features of the input big data
y	responses of the input big data
lam	robustness parameter takes value between 0 (fully supervised) and 1 (fully unsupervised)
standardize	should the big data be normalized to have zero mean unit variance

Details

The supercompress algorithm finds the n compressed points by sequentially splitting the space into n Voronoi regions with centers being the n compressed points. The splitting is done to minimize the total within-cluster sum of squares. The parameter lam controls the robustness of the splitting, with value 0 being fully supervised (objective based on response y only) and value 1 being fully unsupervised (objective based on feature x only), where the latter case reduces to the kmeans clustering. The Vornoi regions are identified by the fast nearest neighbor search implemented in the R package FNN. Only continuous response and features are supported at this time. Default is to standardize the big data to have zero mean and unit variance before processing. Please see Joseph and Mak (2021) for details.

Value

D	features of compressed data points
ybar	responses of compressed data points
cluster	a vector of integers indicating assignment of each point to its nearest compressed data point
l2	the total sum of squares

Author(s)

Chaofan Huang and V. Roshan Joseph

References

Joseph, V. R. and Mak, S. (2021). Supervised compression of big data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 14(3), 217-229.

Beygelzimer, A., Kakadet, S., Langford, J., Arya, S., Mount, D., and Li, S. (2019). FNN: Fast nearest neighbor search algorithms and applications, R 1.1.3.

Examples

```
#############################################################################
# One dimensional example
#############################################################################
# generate big data
set.seed(1)
N <- 3000
x <- seq(0,1,length=N)
f <- function(x) dnorm(x, mean = 0.4, sd = 0.01)
y <- f(x) + 0.1 * rnorm(N)
x <- matrix(x, ncol=1)
# visualize big data
plot(x,y,cex=.5,main="Big Data",cex.main=3,xlab="x",ylab="y",cex.lab=2, cex.axis=2)
# big data reduction via supercompress
n <- 30
sc <- supercompress(n,x,y,lam=0)
D <- sc$D # reduced data point input features
ybar <- sc$ybar # reduced data point response
points(cbind(D, ybar), pch=4,col=4,lwd=4, cex=1.5)

#############################################################################
# Two dimensional Michaelwicz function
#############################################################################
f=function(x) {
  p=length(x)
  x=pi*x
  val=-sum(sin(x)*(sin((1:p)*x^2/pi))^(20))
  return(val)
}
# generate big data
p=2
N=10000*p
set.seed(1)
x=NULL
for(i in 1:p) x=cbind(x,runif(N))
y=apply(x,1,f)+.0001*rnorm(N)
true=apply(x,1,f)
# groundtruth
N.plot=250
p1=seq(0,1,length=N.plot)
p2=seq(0,1,length=N.plot)
fc=matrix(apply(expand.grid(p1,p2),1,f),nrow = N.plot, ncol= N.plot)
# big data reduction via supercompress
n <- 100
sc <- supercompress(n,x,y,lam=1/(1+p))
D <- sc$D # reduced data point input features
ybar <- sc$ybar # reduced data point response
image(p1,p2,fc,col=cm.colors(5),xlab=expression(x[1]),ylab=expression(x[2]),
main="robust-supervised",cex.main=3,cex.lab=2, cex.axis=2)
points(D,pch=16,col=4,cex=2)
```

Index