Package: scR (via r-universe)

December 18, 2024

Title Estimate Vapnik-Chervonenkis Dimension and Sample Complexity

Version 0.3.0

Description We provide a suite of tools for estimating the sample complexity of a chosen model through theoretical bounds and simulation. The package incorporates methods for estimating the Vapnik-Chervonenkis dimension (VCD) of a chosen algorithm, which can be used to estimate its sample complexity. Alternatively, we provide simulation methods to estimate sample complexity directly. For more details, see Carter, P & Choi, D (2024). ``Learning from Noise: Applying Sample Complexity for Political Science Research" <doi:10.31219/osf.io/evrcj>.

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/pjesscarter/scR

BugReports https://github.com/pjesscarter/scR/issues

Imports parallel, pbapply, caret, dplyr, tidyr, ggplot2, plotly

Suggests rmarkdown

Depends R (>= 2.10)

LazyData true

LazyDataCompression xz

License MIT + file LICENSE

NeedsCompilation no

Author Perry Carter [aut, cre] (<https://orcid.org/0000-0002-4684-6533>), Dahyun Choi [aut] (<https://orcid.org/0000-0002-2628-1467>)

Maintainer Perry Carter <pjc504@nyu.edu>

Repository CRAN

Date/Publication 2024-12-18 16:30:02 UTC

Config/pak/sysreqs make libicu-dev libssl-dev

Contents

`acc_sim` *Utility function to generate accuracy metrics, for use with* `estimate_accuracy()`

Description

Utility function to generate accuracy metrics, for use with `estimate_accuracy()`

Usage

```
acc_sim(n, method = "Uniform", p = NULL, ...)
```

Arguments

`n`	An integer giving the desired sample size for which the target function is to be calculated.
`method`	An optional string stating the distribution from which data is to be generated. Default is i.i.d. uniform sampling. Currently also supports "Class Imbalance". Can also take a function outputting a vector of probabilities if the user wishes to specify a custom distribution.
`p`	If method is 'Class Imbalance', gives the degree of weight placed on the positive class.
`...`	Additional model parameters to be specified by the user.

Value

A data frame giving performance metrics for the specified sample size.

br *Replication data for 'Predicting Recidivism'*

Description

Replication data for 'Predicting Recidivism'

Usage

```
br
```

Format

An object of class `data.frame` with 7214 rows and 14 columns.

Author(s)

Julia Dressel and Hany Farid

References

https://www.science.org/doi/full/10.1126/sciadv.aao5580

estimate_accuracy *Estimate sample complexity bounds for a binary classification algorithm using either simulated or user-supplied data.*

Description

Estimate sample complexity bounds for a binary classification algorithm using either simulated or user-supplied data.

Usage

```
estimate_accuracy(
  formula,
  model,
  data = NULL,
  dim = NULL,
  maxn = NULL,
  upperlimit = NULL,
  nsample = 30,
  steps = 50,
  eta = 0.05,
  delta = 0.05,
  epsilon = 0.05,
```

```
  predictfn = NULL,
  power = FALSE,
  effect_size = NULL,
  powersims = NULL,
  alpha = 0.05,
  parallel = TRUE,
  coreoffset = 0,
  packages = list(),
  method = c("Uniform", "Class Imbalance"),
  p = NULL,
  minn = ifelse(is.null(data), (dim + 1), (ncol(data) + 1)),
  x = NULL,
  y = NULL,
  ...
)
```

Arguments

formula	A formula that can be passed to the model argument to define the classification algorithm
model	A binary classification model supplied by the user. Must take arguments formula and data
data	Optional. A rectangular data.frame object giving the full data from which samples are to be drawn. If left unspecified, gendata() is called to produce synthetic data with an appropriate structure.
dim	Required if data is unspecified. Gives the horizontal dimension of the data (number of predictor variables) to be generated.
maxn	Required if data is unspecified. Gives the vertical dimension of the data (number of observations) to be generated.
upperlimit	Optional. A positive integer giving the maximum sample size to be simulated, if data was supplied.
nsample	A positive integer giving the number of samples to be generated for each value of n. Larger values give more accurate results.
steps	A positive integer giving the number of values of n for which simulations should be conducted. Larger values give more accurate results.
eta	A real number between 0 and 1 giving the probability of misclassification error in the training data.
delta	A real number between 0 and 1 giving the targeted maximum probability of observing an OOS error rate higher than epsilon
epsilon	A real number between 0 and 1 giving the targeted maximum out-of-sample (OOS) error rate
predictfn	An optional user-defined function giving a custom predict method. If also using a user-defined model, the model should output an object of class "svrclass" to avoid errors.
power	A logical indicating whether experimental power based on the predictions should also be reported

effect_size	If power is TRUE, a real number indicating the scaled effect size the user would like to be able to detect.
powersims	If power is TRUE, an integer indicating the number of simulations to be conducted at each step to calculate power.
alpha	If power is TRUE, a real number between 0 and 1 indicating the probability of Type I error to be used for hypothesis testing. Default is 0.05.
parallel	Boolean indicating whether or not to use parallel processing.
coreoffset	If parallel is true, a positive integer indicating the number of free threads to be kept unused. Should not be larger than the number of CPU cores.
packages	A list of packages that need to be loaded in order to run model.
method	An optional string stating the distribution from which data is to be generated. Default is i.i.d. uniform sampling. Can also take a function outputting a vector of probabilities if the user wishes to specify a custom distribution.
p	If method is 'Class Imbalance', gives the degree of weight placed on the positive class.
minn	Optional argument to set a different minimum n than the dimension of the algorithm. Useful with e.g. regularized regression models such as elastic net.
x	Optional argument for methods that take separate predictor and outcome data. Specifies a matrix-like object containing predictors. Note that if used, the x and y objects are bound together columnwise; this must be handled in the user-supplied helper function.
y	Optional argument for methods that take separate predictor and outcome data. Specifies a vector-like object containing outcome values. Note that if used, the x and y objects are bound together columnwise; this must be handled in the user-supplied helper function.
...	Additional arguments that need to be passed to model

Value

A list containing two named elements. Raw gives the exact output of the simulations, while Summary gives a table of accuracy metrics, including the achieved levels of ϵ and δ given the specified values. Alternative values can be calculated using getpac()

See Also

plot_accuracy(), to represent simulations visually, getpac(), to calculate summaries for alternate values of ϵ and δ without conducting a new simulation, and gendata(), to generated synthetic datasets.

Examples

```
mylogit <- function(formula, data){
m <- structure(
  glm(formula=formula,data=data,family=binomial(link="logit")),
  class=c("svrclass","glm")  #IMPORTANT - must use the class svrclass to work correctly
)
```

```
return(m)
}
mypred <- function(m,newdata){
out <- predict.glm(m,newdata,type="response")
out <- factor(ifelse(out>0.5,1,0),levels=c("0","1"))
#Important - must specify levels to account for possibility of all
#observations being classified into the same class in smaller samples
return(out)
}

library(parallel)
  results <- estimate_accuracy(two_year_recid ~
    race + sex + age + juv_fel_count + juv_misd_count + priors_count +
    charge_degree..misd.fel.,mylogit,br,
    predictfn = mypred,
    nsample=10,
    steps=10,
    coreoffset = (detectCores() -2)
  )
```

---

gendata *Simulate data with appropriate structure to be used in estimating sample complexity bounds*

Description

Simulate data with appropriate structure to be used in estimating sample complexity bounds

Usage

```
gendata(model, dim, maxn, predictfn = NULL, varnames = NULL, ...)
```

Arguments

model	A binary classification model supplied by the user. Must take arguments `formula` and `data`
dim	Gives the horizontal dimension of the data (number of predictor variables) to be generated.
maxn	Gives the vertical dimension of the data (number of observations) to be generated.
predictfn	An optional user-defined function giving a custom predict method. If also using a user-defined model, the `model` should output an object of class `"svrclass"` to avoid errors.
varnames	An optional character vector giving the names of variables to be used for the generated data
...	Additional arguments that need to be passed to `model`

Value

A `data.frame` containing the simulated data.

See Also

`estimate_accuracy()`, to estimate sample complexity bounds given the generated data

Examples

```
mylogit <- function(formula, data){
m <- structure(
  glm(formula=formula,data=data,family=binomial(link="logit")),
  class=c("svrclass","glm")  #IMPORTANT - must use the class svrclass to work correctly
)
return(m)
}
mypred <- function(m,newdata){
out <- predict.glm(m,newdata,type="response")
out <- factor(ifelse(out>0.5,1,0),levels=c("0","1"))
#Important - must specify levels to account for possibility of all
#observations being classified into the same class in smaller samples
return(out)
}
formula <- two_year_recid ~
  race + sex + age + juv_fel_count +
  juv_misd_count + priors_count + charge_degree..misd.fel.
dat <- gendata(mylogit,7,7214,mypred,all.vars(formula))

library(parallel)
results <- estimate_accuracy(formula,mylogit,dat,predictfn = mypred,
    nsample=10,
    steps=10,
    coreoffset = (detectCores() -2))
```

---

getpac *Recalculate achieved sample complexity bounds given different parameter inputs*

Description

Recalculate achieved sample complexity bounds given different parameter inputs

Usage

```
getpac(table, epsilon = 0.05, delta = 0.05)
```

Arguments

table	A list containing an element named Raw. Should always be used with the output of estimate_accuracy()
epsilon	A real number between 0 and 1 giving the targeted maximum out-of-sample (OOS) error rate
delta	A real number between 0 and 1 giving the targeted maximum probability of observing an OOS error rate higher than epsilon

Value

A list containing two named elements. Raw gives the exact output of the simulations, while Summary gives a table of accuracy metrics, including the achieved levels of ϵ and δ given the specified values. Alternative values can be calculated using getpac() again.

See Also

plot_accuracy(), to represent simulations visually, getpac(), to calculate summaries for alternate values of ϵ and δ without conducting a new simulation, and gendata(), to generated synthetic datasets.

Examples

```
mylogit <- function(formula, data){
m <- structure(
  glm(formula=formula,data=data,family=binomial(link="logit")),
  class=c("svrclass","glm")  #IMPORTANT - must use the class svrclass to work correctly
)
return(m)
}
mypred <- function(m,newdata){
out <- predict.glm(m,newdata,type="response")
out <- factor(ifelse(out>0.5,1,0),levels=c("0","1"))
#Important - must specify levels to account for possibility of all
#observations being classified into the same class in smaller samples
return(out)
}

library(parallel)
results <- estimate_accuracy(two_year_recid ~ race +
    sex + age + juv_fel_count + juv_misd_count + priors_count +
    charge_degree..misd.fel.,mylogit,br,predictfn = mypred,
    nsample=10,
    steps=10,
    coreoffset = (detectCores() -2))
resultsalt <- getpac(results,epsilon=0.5,delta=0.3)
print(resultsalt$Summary)
```

loss — *Utility function to define the least-squares loss function to be optimized for* simvcd()

Description

Utility function to define the least-squares loss function to be optimized for simvcd()

Usage

```
loss(h, ngrid, xi, a = 0.16, a1 = 1.2, a11 = 0.14927)
```

Arguments

h	A positive real number giving the current guess at VC dimension
ngrid	Vector of sample sizes for which the bounding function is estimated.
xi	Vector of estimated values of the bounding function, usually obtained from risk_bounds()
a	Scaling coefficient for the bounding function. Defaults to the value given by Vapnik, Levin and Le Cun 1994.
a1	Scaling coefficient for the bounding function. Defaults to the value given by Vapnik, Levin and Le Cun 1994.
a11	Scaling coefficient for the bounding function. Defaults to the value given by Vapnik, Levin and Le Cun 1994.

Value

A real number giving the estimated value of the MSE given the current guess.

See Also

simvcd(), the user-facing function for simulating VC dimension and risk_bounds() to generate estimates for xi.

plot_accuracy — *Represent simulated sample complexity bounds graphically*

Description

Represent simulated sample complexity bounds graphically

Usage

```
plot_accuracy(
  table,
 metrics = c("Accuracy", "Precision", "Recall", "Fscore", "Delta", "Epsilon", "Power"),
  plottype = c("ggplot", "plotly"),
  letters = c("greek", "latin")
)
```

Arguments

table	A list containing an element named Raw. Should always be used with the output of estimate_accuracy()
metrics	A character vector containing the metrics to display in the plot. Can be any of "Accuracy", "Precision", "Recall", "Fscore", "delta", "epsilon"
plottype	A string giving the graphics package to be used to generate the plot. Can be one of "ggplot" or "plotly"
letters	A string determining whether delta and epsilon should be given as greek letters in the plot legend. Defaults to Greek lettering but available in case of rendering issues.

Value

Either a ggplot or plot_ly plot object, depending on the chosen option of plottype.

See Also

estimate_accuracy(), to generate estimated sample complexity bounds.

Examples

```
mylogit <- function(formula, data){
m <- structure(
  glm(formula=formula,data=data,family=binomial(link="logit")),
  class=c("svrclass","glm")  #IMPORTANT - must use the class svrclass to work correctly
)
return(m)
}
mypred <- function(m,newdata){
out <- predict.glm(m,newdata,type="response")
out <- factor(ifelse(out>0.5,1,0),levels=c("0","1"))
#Important - must specify levels to account for possibility of all
#observations being classified into the same class in smaller samples
return(out)
}

library(parallel)
results <- estimate_accuracy(two_year_recid ~ race + sex + age +
      juv_fel_count + juv_misd_count + priors_count +
      charge_degree..misd.fel.,mylogit,br,predictfn = mypred,
    nsample=10,
```

```
    steps=10,
    coreoffset = (detectCores() -2))

fig <- plot_accuracy(results,letters="latin")
fig
```

`risk_bounds` *Utility function to generate data points for estimation of the VC Dimension of a user-specified binary classification algorithm given a specified sample size.*

Description

Utility function to generate data points for estimation of the VC Dimension of a user-specified binary classification algorithm given a specified sample size.

Usage

```
risk_bounds(x, ...)
```

Arguments

`x`	An integer giving the desired sample size for which the target function is to be approximated.
`...`	Additional model parameters to be specified by the user.

Value

A real number giving the estimated value of Xi(n), the bounding function

---

`scb` *Calculate sample complexity bounds for a classifier given target accuracy*

Description

Calculate sample complexity bounds for a classifier given target accuracy

Usage

```
scb(vcd = NULL, epsilon = NULL, delta = NULL, eta = NULL, theor = TRUE, ...)
```

Arguments

vcd	The Vapnik-Chervonenkis dimension (VCD) of the chosen classifier. If theor is FALSE, this can be left unspecified and simvcd() will be called to estimate the VCD
epsilon	A real number between 0 and 1 giving the targeted maximum out-of-sample (OOS) error rate
delta	A real number between 0 and 1 giving the targeted maximum probability of observing an OOS error rate higher than epsilon
eta	A real number between 0 and 1 giving the probability of misclassification error in the training data.
theor	A Boolean indicating whether the theoretical VCD is to be used. If FALSE, it will instead be estimated using simvcd()
...	Arguments to be passed to simvcd()

Value

A real number giving the sample complexity bound for the specified parameters.

See Also

simvcd(), to calculate VCD for a chosen model

Examples

```
mylogit <- function(formula, data){
m <- structure(
  glm(formula=formula,data=data,family=binomial(link="logit")),
  class=c("svrclass","glm")  #IMPORTANT - must use the class svrclass to work correctly
)
return(m)
}
mypred <- function(m,newdata){
out <- predict.glm(m,newdata,type="response")
out <- factor(ifelse(out>0.5,1,0),levels=c("0","1"))
#Important - must specify levels to account for possibility of all
#observations being classified into the same class in smaller samples
return(out)
}
library(parallel)
scb(epsilon=0.05,delta=0.05,eta=0.05,theor=FALSE,
model=mylogit,dim=7,m=10,k=10,maxn=50,predictfn = mypred,
    coreoffset = (detectCores() -2))
vcd <- 7
scb(vcd,epsilon=0.05,delta=0.05,eta=0.05)
```

simvcd *Estimate the Vapnik-Chervonenkis (VC) dimension of an arbitrary binary classification algorithm.*

Description

Estimate the Vapnik-Chervonenkis (VC) dimension of an arbitrary binary classification algorithm.

Usage

```
simvcd(
  model,
  dim,
  packages = list(),
  m = 1000,
  k = 1000,
  maxn = 5000,
  parallel = TRUE,
  coreoffset = 0,
  predictfn = NULL,
  a = 0.16,
  a1 = 1.2,
  a11 = 0.14927,
  ...
)
```

Arguments

model	A binary classification model supplied by the user. Must take arguments `formula` and `data`
dim	A positive integer giving dimension (number of input features) of the model.
packages	A `list` of strings giving the names of packages to be loaded in order to estimate the model.
m	A positive integer giving the number of simulations to be performed at each design point (sample size value). Higher values give more accurate results but increase computation time.
k	A positive integer giving the number of design points (sample size values) for which the bounding function is to be estimated. Higher values give more accurate results but increase computation time.
maxn	Gives the vertical dimension of the data (number of observations) to be generated.
parallel	Boolean indicating whether or not to use parallel processing.
coreoffset	If `parallel` is true, a positive integer indicating the number of free threads to be kept unused. Should not be larger than the number of CPU cores.

predictfn	An optional user-defined function giving a custom predict method. If also using a user-defined model, the model should output an object of class "svrclass" to avoid errors.
a	Scaling coefficient for the bounding function. Defaults to the value given by Vapnik, Levin and Le Cun 1994.
a1	Scaling coefficient for the bounding function. Defaults to the value given by Vapnik, Levin and Le Cun 1994.
a11	Scaling coefficient for the bounding function. Defaults to the value given by Vapnik, Levin and Le Cun 1994.
...	Additional arguments that need to be passed to model

Value

A real number giving the estimated value of the VC dimension of the supplied model.

See Also

scb(), to calculate sample complexity bounds given estimated VCD.

Examples

```
mylogit <- function(formula, data){
m <- structure(
  glm(formula=formula,data=data,family=binomial(link="logit")),
  class=c("svrclass","glm")  #IMPORTANT - must use the class svrclass to work correctly
)
return(m)
}
mypred <- function(m,newdata){
out <- predict.glm(m,newdata,type="response")
out <- factor(ifelse(out>0.5,1,0),levels=c("0","1"))
#Important - must specify levels to account for possibility of all
#observations being classified into the same class in smaller samples
return(out)
}
library(parallel)
vcd <- simvcd(model=mylogit,dim=7,m=10,k=10,maxn=50,predictfn = mypred,
    coreoffset = (detectCores() -2))
```

Index