Package ‘LKT’

July 15, 2022

Title Logistic Knowledge Tracing
Version 1.2.0
Description Computes Logistic Knowledge Tracing ('LKT') which is a general method for tracking human learning in an educational software system. Please see Pavlik, Eglington, and Harrel-Williams (2021) <https://ieeexplore.ieee.org/document/9616435>. 'LKT' is a method to compute features of student data that are used as predictors of subsequent performance. 'LKT' allows great flexibility in the choice of predictive components and features computed for these predictive components. The system is built on top of 'LiblineaR', which enables extremely fast solutions compared to base glm() in R.

License GPL-3
Encoding UTF-8
LazyData true
VignetteBuilder knitr
RoxygenNote 7.2.0
Depends R (>= 3.5.0), SparseM (>= 1.78), methods, Matrix, data.table (>= 1.13.2), LiblineaR (>= 2.10-8), HDInterval (>= 0.2.2)
Imports glmnet (>= 4.0-2), glmnetUtils (>= 1.1.8), lme4 (>= 1.1-23), cluster (>= 2.1.3)
Suggests rmarkdown, pROC (>= 1.16.2), knitr, utils, caret, ggplot2
NeedsCompilation no
Author Philip I. Pavlik Jr. [aut, ctb, cre]
<https://orcid.org/0000-0001-6467-9452>, Luke G. Eglington [aut, ctb] (<https://orcid.org/0000-0002-8432-9203>)
Maintainer Philip I. Pavlik Jr. <imrryr@gmail.com>
Repository CRAN
Date/Publication 2022-07-15 13:00:12 UTC

R topics documented:

  computefeatures .................................................. 2
computefeatures

computeSpacingPredictors ........................................ 3
countOutcome ...................................................... 3
largerawsample ..................................................... 4
LKT ................................................................. 4
LKT_HDI ............................................................ 7
samplelkt ............................................................. 8
smallSet ............................................................. 9
ViewExcel ........................................................... 9

Index

10

Description

Compute feature describing prior practice effect.

Usage

computefeatures(data, feat, par1, par2, index, index2, par3, par4, par5, fcomp)

Arguments

data copy of main data frame.
feat is the feature to be computed.
par1 nonlinear parameters used for nonlinear features.
par2 nonlinear parameters used for nonlinear features.
index a student by component levels index
index2 a component levels index
par3 nonlinear parameters used for nonlinear features.
par4 nonlinear parameters used for nonlinear features.
par5 nonlinear parameters used for nonlinear features.
fcomp the component name.

Value

a vector suitable for regression input.
computeSpacingPredictors

**Description**

Compute repetition spacing time based features from input data CF.Time. and/or CF.reftime. which will be automatically computed from Duration..sec. if not present themselves.

**Usage**

computeSpacingPredictors(data, KCs)

**Arguments**

- `data` is a dataset with Anon.Student.Id and CF.ansbin.
- `KCs` are the components for which spaced features will be specified in LKT

**Value**

data which is the same frame with the added spacing relevant columns.

---

countOutcome

countOutcome

**Description**

Compute the prior sum of the response appearing in the outcome column for the index

**Usage**

countOutcome(data, index, response)

**Arguments**

- `data` the dataset to compute an outcome vector for
- `index` the subsets to count over
- `response` the actually response value being counted

**Value**

the vector of the lagged cumulative sum.
largerawsample

Trial sequences for practice participants.

Description

A dataset containing a raw sample from the Memphis Datashop.

Usage

largerawsample

Format

A data frame please see the DataShop for more info.
It has many columns.

Source

https://datashop.memphis.edu/Export?datasetId=1465

LKT

Description

Compute a logistic regression model of learning for input data.

Usage

LKT(
data, components, features,
fixedpars = NA, seedpars = NA,
covariates = NA, curvefeats = NA,
dualfit = FALSE, interc = FALSE,
cv = FALSE, elastic = FALSE,
verbose = TRUE, epsilon = 1e-04,
cost = 512, lowb = 1e-05,
highb = 0.99999,
type = 0,
makeTimes = FALSE,
bias = 0,
maxitv = 100,
autoKC = rep(0, length(components)),
autoKCcont = rep("NA", length(components)),
connectors = rep("+", length(components) - 1)
)

Arguments

data  A dataset with Anon.Student.Id and CF..ansbin.
components  A vector of factors that can be used to compute each features for each subject.
features  a vector methods to use to compute a feature for the component.
fixedpars  a vector of parameters for all features+components.
seedpars  a vector of parameters for all features+components to seed non-linear parameter search.
covariates  A list of components that interacts with component by feature in the main specification.
curvefeats  vector of columns to use with "diff" functions
dualfit  TRUE or FALSE, fit a simple latency using logit. Requires Duration..sec. column in data.
interc  TRUE or FALSE, include a global intercept.
cv  TRUE or FALSE, if TRUE runs N-fold cv. Requires premade column named 'fold' with integers denoting the N folds
elastic  glmnet, cv.glmnet, cva.glmnet or FALSE.
verbose  provides more output in some cases.
epsilon  passed to LiblineaR
cost  passed to LiblineaR
lowb  lower bound for non-linear optimizations
highb  upper bound for non-linear optimizations
type  passed to LiblineaR
makeTimes  Boolean indicating whether to create time based features (or may be precomputed)
bias  passed to LiblineaR
maxitv  passed to nonlinear optimization a maxit control
autoKC  a vector to indicate whether to use autoKC for the component (0) or the k for the number of clusters
autoKCcont  a vector of text strings set to "rand" for component to make autoKC assignment to cluster is randomized (for comparison)
connectors  a vector if linear equation R operators including +, * and :
Value

list of values "model", "coefs", "r2", "prediction", "nullmodel", "latencymodel", "optimizedpars", "subjectrmse", "newdata", and "automat"

Examples

temp <- samplelkt
temp$CF..ansbin. <- ifelse(temp$Outcome == "CORRECT", 1, ifelse(temp$Outcome == "INCORRECT", 0, -1))
temp <- data.table::setDT(temp)
temp <- computeSpacingPredictors(temp, "KC..Default.")
temp <- temp[temp$CF..ansbin == 0 | temp$CF..ansbin. == 1,]
temp$KC..Default. <- substr(temp$KC..Default., 1, 10)
modelob <- LKT(
data = temp, interc=TRUE,
components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
features = c("logitdec", "logitdec", "lineafm"),
fixedpars = c(.9, .85)
)
print(modelob$coefs)
print(modelob$loglik)

modelob <- LKT(
data = temp, interc=TRUE,
components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
features = c("logitdec", "logitdec", "lineafm"),
seedpars = c(.9, .85)
)
print(modelob$coefs)
print(modelob$loglik)

modelob <- LKT(
data = temp, interc=TRUE,
connectors = c("+","x","+")
components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
features = c("intercept", "logitdec", "lineafm"),
fixedpars = c(.9, .85)
)
print(modelob$coefs)
print(modelob$loglik)

modelob <- LKT(
data = temp, interc=TRUE,
components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
features = c("logitdec", "logitdec$", "lineafm$"),
fixedpars = c(.9, .85)
)
print(modelob$coefs)
print(modelob$loglik)

# this example illustrates how mean fit is worse for CV
# compared to the first example above. In this case,
# this is mainly do to the small dataset allowing overgeneralization
# despite the model only having 4 coefficients
temp <- samplelkt
unq <- sample(unique(temp$Anon.Student.Id))
sfold <- rep(1:5,length.out=length(unq))
temp$fold <- rep(0,length(temp[,1]))
for(i in 1:5) {temp$fold[which(temp$Anon.Student.Id %in% unq[which(sfold==i)])] = i}
modelob <- LKT(
    data = temp, interc=TRUE,
    components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
    features = c("logitdec", "logitdec", "lineafm"),
    fixedpars = c(.9, .85), cv=TRUE
)
print(modelob$cv_res)
print(mean(modelob$cv_res$rmse))
print(mean(modelob$cv_res$mcfad))

# this example illustrates the limitation of CV when data does not contain
# sufficient examples of each predictor
#modelob <- LKT(
#    data = temp, interc=TRUE,
#    components = c("Anon.Student.Id", "KC..Default.", "KC..Default."),
#    features = c("logitdec", "logitdec", "lineafm"),
#    fixedpars = c(.9, .85), cv=TRUE
#)
# print(modelob$cv_res)

LKT_HDI

Description

Bootstrap credibility intervals to aid in interpreting coefficients.

Usage

LKT_HDI(
    dat,
    n_boot,
    n_students,
    components,
    features,
    covariates,
    fixedpars,
    get_hdi = TRUE,
    cred_mass = 0.95
)
Arguments

dat  Dataframe
n_boot  Number of subsamples to fit
n_students  Number of students per subsample
components  components in model
features  features in model
covariates  covariates in model
fixedpars  fixed pars in model
get_hdi  boolean to decide if generating HDI per coefficient
cred_mass  credibility mass parameter to decide width of HDI

Value

list of values "par_reps", "mod_full", "coef_hdi"

```
samplelkt  Trial sequences for practice participants.
```

Description

A dataset containing a small sample of participants in a memory experiment.

Usage

samplelkt

Format

A data frame with 2074 rows and many variables:

Anon.Student.Id  unique identifier for each student
Duration..sec.  unique identifier for each student
KC..Default.  unique identifier for each student
Outcome  unique identifier for each student ...

Source

https://datashop.memphis.edu/index.jsp
**smallSet**

---

**smallSet**

---

**Description**

smallSet

**Usage**

smallSet(data, nSub)

**Arguments**

- data: Dataframe of student data
- nSub: Number of students

---

**ViewExcel**

---

**Description**

ViewExcel

**Usage**

ViewExcel(df = .Last.value, file = tempfile(fileext = ".csv"))

**Arguments**

- df: Dataframe
- file: Name of the Excel file
Index

* datasets
  largerawsample, 4
  samplelkt, 8

computefeatures, 2
computeSpacingPredictors, 3
countOutcome, 3

largerawsample, 4
LKT, 4
LKT_HDI, 7

samplelkt, 8
smallSet, 9

ViewExcel, 9