Package 'TLBC'

August 29, 2016

Type Package

Title Two-Level Behavior Classification

Version 1.0

Date 2015-10-08

Author Katherine Ellis

Maintainer Katherine Ellis <kellis@ucsd.edu>

Description Contains functions for training and applying two-level random forest and hidden Markov models for human behavior classification from raw tri-axial accelerometer and/or GPS data. Includes functions for training a two-level model, applying the model to data, and computing performance.

Depends R(>= 2.10)

Imports stringr,randomForest,HMM,tools,signal,caret

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2015-10-14 18:13:22

R topics documented:

TLBC-package 2
alignStart 4
annotationsToLabels 5
calcPerformance 5
classify 6
clearFiles 7
computeEmissionProbs 8
computeOneAccFeat 9
computeOneGPSFeat 9
computePriorProbs 10
computeTransProbs 11
distance 11
extractAccelerometerFeatures 12

extractAccFeatsFile . . . 13
extractFeatsPALMSDir . . . 13
extractFeatsPALMSOneFile . . . 14
extractLabelsDir . . . 15
extractLabelsSingleFile . . . 15
getDateFmt . . . 16
hmm . . . 16
isFeatureDirectory . . . 17
isInstanceFormat . . . 17
loadData . . . 18
loadFeatures . . . 18
loadLabels . . . 19
loadModel . . . 20
loadPredictions . . . 20
loadPredictionsAndLabels . . . 21
looXval . . . 22
rf . . . 23
senseCamLabels . . . 23
sensorsToFeatures . . . 24
stratSample . . . 25
testHMM . . . 25
testRF . . . 26
testTwoRFs . . . 27
trainHMM . . . 27
trainModel . . . 28
trainRF . . . 30
winSize . . . 31
writePredictions . . . 31

Index **32**

`TLBC-package` *Two-Level Behavior Classification*

Description

Contains functions for training and applying two-level random forest and hidden Markov models for human behavior classification from raw tri-axial accelerometer and/or GPS data.

This code works with csv data from Actigraph accelerometers (please export in RAW format, without timestamps), and/or with GPS data processed by the PALMS GPS cleaning software.

The TLBC classifier uses six behavior labels:

- Sitting
- Standing Still
- Standing Moving
- Walking/Running

- Bicycling
- Vehicle

Function classify uses a pre-learned TLBC model to classify accelerometer and/or GPS data with behavior labels. Pre-trained models that have been trained on three UCSD datasets are available for download.

Function trainModel trains a TLBC model from annotated accelerometer and/or GPS data.

Function calcPerformance computes the accuracy of predictions made on a given dataset.

Function looXval performs leave-one-out cross-validation on a dataset.

Details

Package:	TLBC
Type:	Package
Version:	1.0
Date:	2015-05-29
License:	GPL-2

Author(s)

Katherine Ellis <kellis@ucsd.edu>

See Also

randomForest, HMM

Examples

```
## Not run:

# train a new model
myAnnotations="~/myStudy/annotations.csv"
myAccel="~/myStudy/HipGT3X+"
myGPS="~/myStudy/GPS.csv"
WS=60
myModel="~/myStudy/myModel.RData"
trainModel(annotations=myAnnotations, accelerometers=myAccel, GPS=myGPS, winSize=WS,
modelName=myModel)

# classify using a model computed yourself
myAccel="~/myStudy/HipGT3X+"
myGPS="~/myStudy/GPS.csv"
myModel="~/myStudy/myModel.RData"
myPredictions="~/myStudy/myModelPredictions"
classify(accelerometers=myAccel, GPS=myGPS, modelName=myModel, saveDir=myPredictions)
```

```
# compute the performance of a model on a dataset
myAnnotations="~/myStudy/annotations.csv"
myPredictions="~/myStudy/myModelPredictions"
WS=60
calcPerformance(annotations=myAnnotations, predictions=myPredictions, winSize=WS)

# perform leave-one-out cross-validation on a dataset
myAnnotations="~/myStudy/annotations.csv"
myAccel="~/myStudy/HipGT3X+"
WS=60
myPredictions="~/myStudy/looXvalPredictions"
looXval(annotations=myAnnotations, accelerometers=myAccel, winSize=WS, saveDir=myPredictions)


## End(Not run)
```

`alignStart` *Function to align start of a window*

Description

Aligns start time to nearest minute interval.

Usage

```
alignStart(winSize, start)
```

Arguments

`winSize`	Window size in seconds.
`start`	The start time as a POSIXlt DateTime object.

Author(s)

Katherine Ellis

See Also

`DateTimeClasses`

annotationsToLabels *Function to convert bout-level annotations to instance-level labels*

Description

Converts bout-level annotations to instance-level labels.

Usage

```
annotationsToLabels(annotations, winSize, names = NULL)
```

Arguments

annotations	Path to file containing bout-level annotations, or directory of files containing bout-level annotations. Should be csv format with fields: *identifier, StartDateTime, EndDateTime, behavior*.
winSize	Window size in seconds.
names	(Optional) If provided, extract annotations only for identifiers in this list.

Value

Path to directory where instance-level label files are saved.

Author(s)

Katherine Ellis

calcPerformance *Function to calculate performance of a classification model*

Description

Calculates several performance metrics.

Usage

```
calcPerformance(annotations, predictions, winSize, names=NULL, combineStanding=FALSE)
```

Arguments

annotations	Path to file containing bout-level annotations, a directory of files containing bout-level annotations, or a directory of instance-level annotations (*i.e.*, the output of funciton *annotationsToLabels*).
predictions	Path to directory containing predictions (*i.e.*, the *saveDir* argument to the function *classify*).
winSize	Window size in seconds.
names	(Optional) List of identifiers to use.
combineStanding	logical: combine standing still and standing moving into a single category?

Value

Object containing confusion matrix and several performance metrics.

Author(s)

Katherine Ellis

See Also

confusionMatrix

Examples

```
## Not run:

# compute the performance of a model on a dataset
myAnnotations="~/myStudy/annotations.csv"
myPredictions="~/myStudy/myModelPredictions"
WS=60
calcPerformance(annotations=myAnnotations, predictions=myPredictions, winSize=WS)

## End(Not run)
```

---

classify *Function to classify accelerometer and/or GPS data*

Description

Classifies data into behavior categories using a pre-computed two-level model.

Usage

```
classify(accelerometers=NULL, GPS=NULL, modelName, saveDir, names=NULL)
```

Arguments

accelerometers	(Optional) Path to a directory (or list of directories) containing actigraph accelerometer data files. Accelerometer data files should be csv files output in "raw" format by ActiLife (without timestamps), and named by the participant identifier, *e.g.*, Participant01.csv. Or, path to a directory (or list of directories) containing previously computed accelerometer features, *i.e.*, computed by the function *sensorsToFeatures*.
GPS	(Optional) Path to a PALMS-processed GPS data file (or a directory containing GPS data files). GPS data files should be in csv format with the following fields: *identifier, dateTime, speed, ele, elevationDelta, lat, lon, nsatView, snrView*. *identifier* should be the participant identifier, *e.g.* Participant01. If *GPS* is a path to a directory, each file in the directory should correspond to a participant, and the file name should be the participant identifier, *e.g.*, Participant01.csv. Or, path to a directory containing previously computed GPS features, *i.e.*, computed by the function *sensorsToFeatures*.
modelName	Path to a pre-trained TLBC model. Either a model you trained yourself, *i.e.*, the argument *modelName* in the function *trainModel*, or a pre-trained model that has been trained on one of three UCSD datasets, available for download.
saveDir	Path to a directory where predictions will be saved. Predictions will be saved in files named <*identifier*>.csv with two fields: *timestamp, prediction*.
names	(Optional) List of participant identifiers to use.

Author(s)

Katherine Ellis

Examples

```
## Not run:

# use a pre-trained model to classify hip accelerometer data
myAccel="~/myStudy/HipGT3X+"
ovrWgtModel="OverweightWomenHipGT3X+.RData"
myPredictions="~/myStudy/predictions"
classify(accelerometers=myAccel, modelName=ovrWgtModel, saveDir=myPredictions)

## End(Not run)
```

---

clearFiles *Clear files*

Description

Deletes files in a directory

Usage

```
clearFiles(dir)
```

Arguments

dir	Path to directory to clear.

Author(s)

Katherine Ellis

`computeEmissionProbs` *Compute emission probabilities*

Description

Function to compute the emission probablilities of an HMM, based on the first-level random forest classifier.

Usage

```
computeEmissionProbs(rf)
```

Arguments

rf	A random forest object.

Author(s)

Katherine Ellis

See Also

`trainModel`

computeOneAccFeat *Compute one acceleration feature*

Description

Function to compute one acceleration feature from a data window.

Usage

```
computeOneAccFeat(w, Fs)
```

Arguments

w	N x 3 matrix of 3-axis accelerometer measurements.
Fs	Sample frequency, in Hertz.

Value

Vector of acceleration features.

Author(s)

Katherine Ellis

See Also

extractAccelerometerFeatures

computeOneGPSFeat *Compute one GPS feature*

Description

Function to compute one GPS feature from a data window.

Usage

```
computeOneGPSFeat(w, lastCoordinates)
```

Arguments

w	Data frame of GPS measurements from the PALMS system. Should contain the following fields: *identifier, dateTime, speed, distance, duration, ele, elevationDelta, lat, lon, nsatUsed, nsatView, snrUsed, snrView, fixType*
lastCoordinates	Coordinates of last data sample, to compute change in position - data frame with fields *lat* and *lon*.

Value

Vector of GPS features.

Author(s)

Katherine Ellis

See Also

`extractFeatsPALMSDir`, `extractFeatsPALMSOneFile`

`computePriorProbs` *Compute prior probabilities*

Description

Compute the prior probabilities of an HMM from a state sequence.

Usage

```
computePriorProbs(stateSeq)
```

Arguments

`stateSeq`	Vector of states.

Value

Vector of probabilities for each state.

Author(s)

Katherine Ellis

See Also

`trainModel`

computeTransProbs *Compute transition probabilities*

Description

Compute the transition probabilities of an HMM from a state sequence.

Usage

```
computeTransProbs(stateSeq)
```

Arguments

stateSeq	Vector of states.

Value

Matrix of probabilities for each transition between states.

Author(s)

Katherine Ellis

See Also

trainModel

distance *Distance*

Description

Function to compute the distance between two sets of latitude and longitude coordiantes.

Usage

```
distance(origin, destination)
```

Arguments

origin	A data frame with the the fields *lat* and *lon*.
destination	A data frame with the the fields *lat* and *lon*.

Value

Distance in meters.

Author(s)

Katherine Ellis

See Also

computeOneGPSFeat

`extractAccelerometerFeatures` *Extract accelerometer features*

Description

Function to extract accelerometer features for all GT3X+ raw data files in a directory.

Usage

```
extractAccelerometerFeatures(input, output, winSize, names=NULL)
```

Arguments

`input`	Path to GT3X+ raw data file or a directory containing GT3X+ raw data files.
`output`	Path to a directory to save computed features. A separate directory for each GT3X+ file will be created containing features for each day.
`winSize`	Window size in seconds.
`names`	(Optional) If provided, compute features only for filenames in this list.

Author(s)

Katherine Ellis

See Also

extractAccFeatsFile

extractAccFeatsFile *Extract accelerometer features from a file*

Description

Function to extract accelerometer features from a single GT3X+ raw data file

Usage

```
extractAccFeatsFile(inputFile, outputPath, winSize)
```

Arguments

inputFile	Path to a GT3X+ raw data file.
outputPath	Path to a directory to save computed features. A separate file for each day will be created inside the directory.
winSize	Window size in seconds.

Author(s)

Katherine Ellis

See Also

extractAccelerometerFeatures

extractFeatsPALMSDir *Extract GPS features from a PALMS directory*

Description

Function to extract GPS features from a directory containing PALMS-filtered GPS data files.

Usage

```
extractFeatsPALMSDir(inputDir, outputDir, winSize, names=NULL)
```

Arguments

inputDir	Path to a directory containing PALMS-filtered GPS data files. Each file should be a csv file with the following fields: *identifier, dateTime, speed, distance, duration, ele, elevationDelta, lat, lon, nsatUsed, nsatView, snrUsed, snrView, fixType*.
outputDir	Path to a directory to save computed features. A separate directory for each input file will be created containing features for each day.
winSize	Window size in seconds.
names	(Optional) If provided, compute features only for filenames in this list.

Author(s)

Katherine Ellis

See Also

`extractFeatsPALMSOneFile`, `computeOneGPSFeat`

`extractFeatsPALMSOneFile`

Extract GPS features from a PALMS file

Description

Function to extract GPS features from PALMS-filtered GPS data file.

Usage

```
extractFeatsPALMSOneFile(inputFile, outputDir, winSize, names=NULL)
```

Arguments

`inputFile`	Path to a file containing PALMS-filtered GPS data. Should be a csv file with the following fields: *identifier, dateTime, speed, distance, duration, ele, elevationDelta, lat, lon, nsatUsed, nsatView, snrUsed, snrView, fixType*.
`outputDir`	Path to a directory to save computed features. A separate file will be created containing features for each day.
`winSize`	Window size in seconds.
`names`	(Optional) If provided, compute features only for identifiers in this list.

Author(s)

Katherine Ellis

See Also

`extractFeatsPALMSDir`, `computeOneGPSFeat`

`extractLabelsDir` *Extract labels from a directory*

Description

Function to extract labels from a directory containing annotation files.

Usage

```
extractLabelsDir(inputDir, outputDir, winSize, names=NULL)
```

Arguments

`inputDir`	Path to a directory of files containing bout-level annotations. Should be csv format with fields: *identifier, StartDateTime, EndDateTime, behavior*.
`outputDir`	Path to a directory to save labels.
`winSize`	Window size in seconds.
`names`	(Optional) If provided, extract labels only for identifiers in this list.

Author(s)

Katherine Ellis

See Also

`annotationsToLabels`, `extractLabelsSingleFile`

`extractLabelsSingleFile` *Extract labels from a directory*

Description

Function to extract labels from a directory containing annotation files.

Usage

```
extractLabelsSingleFile(inputFile, outputDir, winSize)
```

Arguments

`inputFile`	Path to a file containing bout-level annotations. Should be csv format with fields: *identifier, StartDateTime, EndDateTime, behavior*.
`outputDir`	Path to a directory to save labels.
`winSize`	Window size in seconds.

Author(s)

Katherine Ellis

See Also

annotationsToLabels, extractLabelsDir

---

getDateFmt *Get date format*

Description

Function to guess the date format from an input string

Usage

```
getDateFmt(inputString)
```

Arguments

inputString	String containing a date formatted either as yyyy-mm-dd HH:MM:SS or mm/dd/yyyy HH:MM:SS

Value

A date format string, either "%Y-%m-%d %H:%M:%S" or "%m/%d/%Y %H:%M:%S"

Author(s)

Katherine Ellis

---

hmm *Hidden Markov model*

Description

Used for loading HMM from a TLBC model.

Author(s)

Katherine Ellis

isFeatureDirectory *Is feature directory?*

Description

Function to check if the directory is a feature directory or a raw data directory (by checking for existence of sub-directories).

Usage

```
isFeatureDirectory(dir)
```

Arguments

dir	Path to a directory.

Value

TRUE, if the directory is a feature directory. *FALSE* otherwise.

Author(s)

Katherine Ellis

isInstanceFormat *Is instance format?*

Description

Function to check if the directory contains instance-level annotations or bout-level annotations (by checking for existence of sub-directories).

Usage

```
isInstanceFormat(annotations)
```

Arguments

annotations	Path to an annotations directory.

Value

TRUE, if the directory contains instance-level annotations. *FALSE* otherwise.

Author(s)

Katherine Ellis

loadData *Load data*

Description

Function to load corresponding label and feature data.

Usage

```
loadData(labelDir, featDirs, names=NULL)
```

Arguments

labelDir	Path to a label directory containing instance-level label files.
featDirs	List of paths to feature directories containing feature files.
names	(Optional) If provided, only load data for identifiers provided in this list.

Value

A list containing (1) a data frame of labels and (2) a data frame of features.

Author(s)

Katherine Ellis

See Also

loadFeatures, loadLabels

loadFeatures *Load features*

Description

Function to load feature data.

Usage

```
loadFeatures(featDirs, names=NULL)
```

Arguments

featDirs	List of paths to feature directories containing feature files.
names	(Optional) If provided, only load data for identifiers provided in this list.

Value

A data frame of features.

Author(s)

Katherine Ellis

See Also

loadData, loadLabels

`loadLabels` *Load labels*

Description

Function to load label data.

Usage

```
loadLabels(labelDir, names=NULL)
```

Arguments

`labelDir`	Path to label directories containing instance-level annotation files.
`names`	(Optional) If provided, only load labels for identifiers provided in this list.

Value

A data frame of labels.

Author(s)

Katherine Ellis

See Also

loadData, loadFeatures

loadModel *Load model*

Description

Function to load a model.

Usage

```
loadModel(modelName, which)
```

Arguments

modelName	Path to model.
which	String specifying which part of model to return. One of "winSize", "rf", or "hmm".

Value

A model (or parameter).

Author(s)

Katherine Ellis

loadPredictions *Load predictions*

Description

Function to load predictions from csv files in a directory.

Usage

```
loadPredictions(predDir, names=NULL)
```

Arguments

predDir	Path to directory containing prediction files.
names	(Optional) If specified, load only predictions for identifiers in this list.

Value

A data frame of predictions.

Author(s)

Katherine Ellis

See Also

loadPredictionsAndLabels

`loadPredictionsAndLabels` *Load predictions and labels*

Description

Function to load corresponding labels and predictions.

Usage

```
loadPredictionsAndLabels(labelDir, predDir, names=NULL)
```

Arguments

`labelDir`	Path to directory containing label files.
`predDir`	Path to directory containing prediction files.
`names`	(Optional) If specified, load only data for identifiers in this list.

Value

A data frame containing predictions and labels.

Author(s)

Katherine Ellis

See Also

loadPredictions

looXval *Function to perform leave-one-out cross-validation*

Description

Performs leave-one-out cross-validation

Usage

```
looXval(annotations, accelerometers=NULL, GPS=NULL, winSize=60, saveDir, names=NULL,
strat=TRUE)
```

Arguments

annotations	Path to a file containing bout-level annotations (or directory of files containing bout-level annotations). Should be csv format with fields: *identifier,StartDateTime,EndDateTime,behavio* Or, path to a directory containing instance-level annotations, *i.e.*, created by the function *annotationsToLabels*.
accelerometers	(Optional) Path to a directory (or list of directories) containing actigraph accelerometer data files. Accelerometer data files should be csv files output in "raw" format by ActiLife (without timestamps), and named by the participant identifier, *e.g.*, Participant01.csv. Or, path to a directory (or list of directories) containing previously computed accelerometer features, *i.e.*, computed by the function *sensorsToFeatures*.
GPS	(Optional) Path to a PALMS-processed GPS data file (or a directory containing GPS data files). GPS data files should be in csv format with the following fields: *identifier, dateTime, speed, ele, elevationDelta, lat, lon, nsatView, snrView*. *identifier* should be the participant identifier, *e.g.* Participant01. If *GPS* is a path to a directory, each file in the directory should correspond to a participant, and the file name should be the participant identifier, *e.g.*, Participant01.csv. Or, path to a directory containing previously computed GPS features, *i.e.*, computed by the function *sensorsToFeatures*.
winSize	Window size in seconds.
saveDir	Path to a directory where predictions will be saved. Predictions will be saved in files named <*identifier*>.csv with two fields: *timestamp,prediction*.
names	(Optional) List of participant identifiers to use.
strat	logical: use stratified sampling for the random forest?

Author(s)

Katherine Ellis

See Also

trainModel,classify

Examples

```
## Not run:

# perform leave-one-out cross-validation on a dataset
myAnnotations="~/myStudy/annotations.csv"
myAccel="~/myStudy/HipGT3X+"
winSize=60
myPredictions="~/myStudy/looXvalPredictions"
looXval(annotations=myAnnotations, accelerometers=myAccel, winSize=WS, saveDir=myPredictions)

## End(Not run)
```

---

rf *Random Forest*

Description

Used for loading RF from a TLBC model.

Author(s)

Katherine Ellis

---

senseCamLabels *SenseCam Labels*

Description

Function to convert bout-level annotations from senseCam labels to 6 categories (can be either single file or directory).

Usage

```
senseCamLabels(input, output)
```

Arguments

input	Path to file or directory containing bout-level annotations.
output	Path to output directory.

Author(s)

Katherine Ellis

`sensorsToFeatures` *Function to extract featurese from raw sensor data*

Description

Extracts features from accelerometer and/or GPS data.

Usage

```
sensorsToFeatures(accelerometers = NULL, GPS = NULL, winSize, names = NULL)
```

Arguments

`accelerometers`	(Optional) Path to a directory (or a list of paths to multiple directories) containing actigraph accelerometer data files. Accelerometer data files should be csv files output in "raw" format by ActiLife (without timestamps), and named by the participant identifier, *e.g.*, Participant01.csv.
`GPS`	(Optional) Path to a PALMS-processed GPS data file (or a directory containing GPS data files). GPS data files should be in csv format with the following fields: *identifier, dateTime, speed, ele, elevationDelta, lat, lon, nsatView, snrView*. *identifier* should be the participant identifier, *e.g.* Participant01. If *GPS* is a path to a directory, each file in the directory should correspond to a participant, and the file name should be the participant identifier, *e.g.*, Participant01.csv.
`winSize`	Window size in seconds.
`names`	(Optional) List of identifiers to use.

Value

List of feature directories created.

Author(s)

Kat Ellis

See Also

`trainModel`

stratSample *Stratified sample*

Description

Function to choose a random sample of data stratified by label.

Usage

```
stratSample(labels, nsamp)
```

Arguments

labels	Vector of strings denoting labels.
nsamp	Number of items to sample from each label class.

Value

Vector of indices chosen.

Author(s)

Katherine Ellis

testHMM *Test a hidden Markov model*

Description

Function to apply a HMM classifier to some data.

Usage

```
testHMM(predDir, modelName, saveDir, names)
```

Arguments

predDir	Path to a directory containing predictions made by the random forest classiifer (*i.e.*, the *saveDir* argument of testRF).
modelName	Path to pre-trained model.
saveDir	Path to a directory where predictions will be saved. Predictions will be saved in files named <*identifier*>.csv with two fields: *timestamp, prediction*.
names	List of participant identifiers to use.

Author(s)

Katherine Ellis

See Also

classify, testRF

---

testRF *Test a random forest classifier*

Description

Function to apply a random forest classifier to some data.

Usage

```
testRF(featDirs, modelName, saveDir, testNames)
```

Arguments

featDirs	Path to a directory (or list of directories) containing features, *i.e.*, computed by the function *sensorsToFeatures*.
modelName	Path to pre-trained model.
saveDir	Path to a directory where predictions will be saved. Predictions will be saved in files named <*identifier*>.csv with two fields: *timestamp, prediction*.
testNames	List of participant identifiers to use.

Author(s)

Katherine Ellis

See Also

classify, testHMM

testTwoRFs *Test two random forest classifiers*

Description

Function to apply two random forest classifiers to some data.

Usage

```
testTwoRFs(featDirs1, featDirs2, rf1, rf2, saveDir, testNames)
```

Arguments

featDirs1	Path to a directory (or list of directories) containing features corresponding to the first RF classifier.
featDirs2	Path to a directory (or list of directories) containing features corresponding to the second RF classifier.
rf1	A random forest model.
rf2	A second random forest model.
saveDir	Path to a directory where predictions will be saved. Predictions will be saved in files named <*identifier*>.csv with two fields: *timestamp, prediction*.
testNames	List of participant identifiers to use.

Author(s)

Katherine Ellis

See Also

testRF

trainHMM *Train a hidden Markov model*

Description

Function to train a HMM classifier from some data and a trained random forest model.

Usage

```
trainHMM(labelDir, rf, names, combineStanding=FALSE)
```

Arguments

labelDir	Path to a directory containing instance-level annotations, *i.e.*, created by the function *annotationsToLabels*.
rf	A random forest model.
names	List of participant identifiers to use.
combineStanding	If TRUE, combine the labels *standing still* and *standing moving* into a single label *standing*.

Author(s)

Katherine Ellis

See Also

trainModel, trainRF

---

trainModel *Function to train a two-level model from accelerometer and/or GPS data*

Description

Trains a two-level model from accelerometer and/or GPS data.

Usage

```
trainModel(annotations, accelerometers=NULL, GPS=NULL, winSize=60, modelName, names=NULL,
strat=TRUE, ntree=500, mtry=NULL, replace=TRUE, nsample=10000, nodesize=1, sampsize=10000)
```

Arguments

annotations	Path to a file containing bout-level annotations (or directory of files containing bout-level annotations). Should be csv format with fields: *identifier,StartDateTime,EndDateTime,behavio* Or, path to a directory containing instance-level annotations, *i.e.*, created by the function *annotationsToLabels*.
accelerometers	(Optional) Path to a directory (or list of directories) containing actigraph accelerometer data files. Accelerometer data files should be csv files output in "raw" format by ActiLife (without timestamps), and named by the participant identifier, *e.g.*, Participant01.csv. Or, path to a directory (or list of directories) containing previously computed accelerometer features, *i.e.*, computed by the function *sensorsToFeatures*.

GPS	(Optional) Path to a PALMS-processed GPS data file (or a directory containing GPS data files). GPS data files should be in csv format with the following fields: *identifier, dateTime, speed, ele, elevationDelta, lat, lon, nsatView, snrView*. *identifier* should be the participant identifier, *e.g.* Participant01. If *GPS* is a path to a directory, each file in the directory should correspond to a participant, and the file name should be the participant identifier, *e.g.*, Participant01.csv. Or, path to a directory containing previously computed GPS features, *i.e.*, computed by the function *sensorsToFeatures*.
winSize	Window size in seconds.
modelName	Path to location to save model.
names	(Optional) List of participant identifiers to use.
strat	(Optional) logical: use stratified sampling for the random forest?
ntree	(Optional) Number of trees in the random forest
mtry	(Optional) Number of variables randomly sampled as candidates at each split in the random forest.
replace	(Optional) Should sampling in the random forest be done with or without replacement?
nsample	(Optional) Number of instances to sample.
nodesize	(Optional) Minimum size of terminal nodes in the random forest.
sampsize	(Optional) Size of sample to draw for the random forest.

Author(s)

Katherine Ellis

See Also

classify, looXval

Examples

```
## Not run:
myAnotations="~/myStudy/annotations.csv"
myAccel="~/myStudy/HipGT3X+"
myGPS="~/myStudy/GPS.csv"
winSize=60
myModel="~/myStudy/myModel.RData"
trainModel(annotations=myAnotations, accelerometers=myAccel, GPS=myGPS, winSize=WS,
modelName=myModel)

## End(Not run)
```

trainRF *Train a random forest classifier*

Description

Function to train a random forest classifier from some data.

Usage

```
trainRF(labelDir, featDirs, names, combineStanding=FALSE, strat=TRUE, ntree=500,
mtry=NULL, replace=TRUE, nsample=10000, nodesize=1, sampsize=10000)
```

Arguments

labelDir	Path to a directory containing instance-level annotations, *i.e.*, created by the function *annotationsToLabels*.
featDirs	Path to a directory (or list of directories) containing features, *i.e.*, computed by the function *sensorsToFeatures*.
names	List of participant identifiers to use.
combineStanding	(Optional) If TRUE, combine the labels *standing still* and *standing moving* into a single label *standing*.
strat	(Optional) logical: use stratified sampling for the random forest?
ntree	(Optional) Number of trees in the random forest
mtry	(Optional) Number of variables randomly sampled as candidates at each split in the random forest.
replace	(Optional) Should sampling in the random forest be done with or without replacement?
nsample	(Optional) Number of instances to sample.
nodesize	(Optional) Minimum size of terminal nodes in the random forest.
sampsize	(Optional) Size of sample to draw for the random forest.

Author(s)

Katherine Ellis

See Also

trainModel, trainHMM

winSize *Window Size*

Description

Used for loading window size from a TLBC model.

Author(s)

Katherine Ellis

writePredictions *Write predictions to a file*

Description

Function to write predictions made by TLBC model to a csv file.

Usage

```
writePredictions(values, timestamps, saveFile)
```

Arguments

values	Vector of predicted labels.
timestamps	Vector of timestamps.
saveFile	Path to file where predictions will be saved.

Author(s)

Katherine Ellis

Index

∗Topic **\textasciitildekwd1**
alignStart, 4
annotationsToLabels, 5
calcPerformance, 5
classify, 6
clearFiles, 7
computeEmissionProbs, 8
computeOneAccFeat, 9
computeOneGPSFeat, 9
computePriorProbs, 10
computeTransProbs, 11
distance, 11
extractAccelerometerFeatures, 12
extractAccFeatsFile, 13
extractFeatsPALMSDir, 13
extractFeatsPALMSOneFile, 14
extractLabelsDir, 15
extractLabelsSingleFile, 15
getDateFmt, 16
hmm, 16
isFeatureDirectory, 17
isInstanceFormat, 17
loadData, 18
loadFeatures, 18
loadLabels, 19
loadModel, 20
loadPredictions, 20
loadPredictionsAndLabels, 21
looXval, 22
rf, 23
senseCamLabels, 23
sensorsToFeatures, 24
stratSample, 25
testHMM, 25
testRF, 26
testTwoRFs, 27
trainHMM, 27
trainModel, 28
trainRF, 30
winSize, 31
writePredictions, 31
∗Topic **\textasciitildekwd2**
alignStart, 4
annotationsToLabels, 5
calcPerformance, 5
classify, 6
clearFiles, 7
computeEmissionProbs, 8
computeOneAccFeat, 9
computeOneGPSFeat, 9
computePriorProbs, 10
computeTransProbs, 11
distance, 11
extractAccelerometerFeatures, 12
extractAccFeatsFile, 13
extractFeatsPALMSDir, 13
extractFeatsPALMSOneFile, 14
extractLabelsDir, 15
extractLabelsSingleFile, 15
getDateFmt, 16
hmm, 16
isFeatureDirectory, 17
isInstanceFormat, 17
loadData, 18
loadFeatures, 18
loadLabels, 19
loadModel, 20
loadPredictions, 20
loadPredictionsAndLabels, 21
looXval, 22
rf, 23
senseCamLabels, 23
sensorsToFeatures, 24
stratSample, 25
testHMM, 25
testRF, 26
testTwoRFs, 27
trainHMM, 27

trainModel, 28
trainRF, 30
winSize, 31
writePredictions, 31
∗Topic **package**
TLBC-package, 2

alignStart, 4
annotationsToLabels, 5, *15, 16*

calcPerformance, *3*, 5
calcPerformanceFromLabels *(*calcPerformance*)*, 5
classify, *3*, 6, *22*, *26*, *29*
clearFiles, 7
computeEmissionProbs, 8
computeOneAccFeat, 9
computeOneGPSFeat, 9, *12*, *14*
computePriorProbs, 10
computeTransProbs, 11
confusionMatrix, *6*

DateTimeClasses, *4*
distance, 11

extractAccelerometerFeatures, *9*, 12, *13*
extractAccFeatsFile, *12*, 13
extractFeatsPALMSDir, *10*, 13, *14*
extractFeatsPALMSOneFile, *10*, *14*, 14
extractLabelsDir, 15, *16*
extractLabelsSingleFile, *15*, 15

getDateFmt, 16

HMM, *3*
hmm, 16

isFeatureDirectory, 17
isInstanceFormat, 17

loadData, 18, *19*
loadFeatures, *18*, 18, *19*
loadLabels, *18, 19*, 19
loadModel, 20
loadPredictions, 20, *21*
loadPredictionsAndLabels, *21*, 21
looXval, *3*, 22, *29*
looXvalFromFeats *(*looXval*)*, 22

randomForest, *3*
rf, 23

senseCamLabels, 23
senseCamLabelsFile *(*senseCamLabels*)*, 23
sensorsToFeatures, 24
stratSample, 25

testAllDir *(*classify*)*, 6
testHMM, 25, *26*
testRF, *25, 26*, 26, *27*
testTwoRFs, 27
TLBC *(*TLBC-package*)*, 2
TLBC-package, 2
trainFromFeatures *(*trainModel*)*, 28
trainHMM, 27, *30*
trainModel, *3*, *8*, *10, 11*, *22*, *24*, *28*, 28, *30*
trainRF, *28*, 30

winSize, 31
writePredictions, 31