Package ‘MLPUGS’

August 29, 2016

Type Package
Title Multi-Label Prediction Using Gibbs Sampling (and Classifier Chains)
Version 0.2.0
Date 2016-07-05
Maintainer Mikhail Popov <mikhail@mpopov.com>
Description An implementation of classifier chains (CC's) for multi-label prediction. Users can employ an external package (e.g. 'randomForest', 'C50'), or supply their own. The package can train a single set of CC's or train an ensemble of CC's -- in parallel if running in a multi-core environment. New observations are classified using a Gibbs sampler since each unobserved label is conditioned on the others. The package includes methods for evaluating the predictions for accuracy and aggregating across iterations and models to produce binary or probabilistic classifications.

URL https://github.com/bearloga/MLPUGS

BugReports https://github.com/bearloga/MLPUGS/issues
Depends R (>= 3.1.2)
Suggests knitr, progress, C50, randomForest
VignetteBuilder knitr
LazyData true
License MIT + file LICENSE
RoxygenNote 5.0.1
NeedsCompilation no
Author Mikhail Popov [aut, cre] (@bearloga on Twitter)
Repository CRAN
Date/Publication 2016-07-06 09:43:54
R topics documented:

MLPUGS-package ................................................. 2
ecc ................................................................. 2
movies ............................................................. 4
predict.ECC ....................................................... 4
summary.PUGS .................................................... 6
validate_pugs ..................................................... 7

Description

An implementation of classifier chains for binary and probabilistic multi-label prediction. The classification pipeline consists of:

1. Training an ensemble of classifier chains. Each chain is a binary classifier (built-in, supplied from an external package or user-coded).
2. Making predictions using a Gibbs sampler since each unobserved label is conditioned on the others.
3. (Optional) Evaluating the ECC.
4. Gathering predictions (aggregating across iterations & models).

To learn more about MLPUGS, start with the vignettes: browseVignettes(package = "MLPUGS")

ecc

Fit an Ensemble of Classifier Chains (ECC)

Description

Constructs an ensemble of classifier chains, each chain using a user-supplied base classifier.

Usage

ecc(x, y, m = 5, prop_subset = 0.95, run_parallel = FALSE, silent = TRUE, .f = NULL, ...)

Arguments

- **x**: A data frame or matrix of features.
- **y**: A data frame or matrix of labels. Each label must be its own column and each instance (observation) must be a row of 0s and 1s, indicating which labels belong to the instance.
- **m**: Number of classifier chains (models) to train. Recommended: $m = 3$ and $m = 7$ for 4-core and 8-core systems, respectively.
- **prop_subset**: The proportion of the training data to utilize when $m$ is greater than 1. Each set of classifier chains in the ensemble will use a random subset (95% by default) of the supplied training data.
- **run_parallel**: Whether to utilize multicore capabilities of the system.
- **silent**: Whether to print progress messages to console. Recommended.
- **.f**: User-supplied classifier training function. If not supplied, the trainer will use the built-in classifier. See Details for more information.
- **...**: additional arguments to pass to `.f`.

Value

An object of class ECC containing:

- **y_labels**: names of the columns of **y**
- **fits**: An list of length $m$, each element being a set of classifier chains - a list of length $L = \text{ncol}(y)$ where each element is a fitted model object trained to predict each of the $L$ labels.

Examples

```r
x <- movies_train[, -(1:3)]
y <- movies_train[, 1:3]
fit <- ecc(x, y, m = 1, .f = glm.fit, family = binomial(link = "logit"))
```

```r
# Not run:
fit <- ecc(x, y, .f = randomForest::randomForest)

fit <- ecc(x, y, m = 7, .f = C50::C5.0, trials = 10)
```

```r
# End(Not run)
```
movies  

FiveThirtyEight's Movie Scores

Description

This dataset contains every film that has a Rotten Tomatoes rating, a RT User rating, a Metacritic score, a Metacritic User score, and IMDb score, and at least 30 fan reviews on Fandango. The data from Fandango was pulled on Aug. 24, 2015. It is licensed under CC BY 4.0

Usage

movies

movies_train

movies_test

Format

A data.frame with 146 rows and 9 columns. The training data contains 87 movies, while the test set contains 59 movies. The first three columns of the training and test sets indicate the multiple labels: 1 if the movie got a rating equal to or greater than 80% on Metacritic, Rotten Tomatoes, and Fandango; and 0 otherwise.

Author(s)

FiveThirtyEight

Source

https://github.com/fivethirtyeight/data/tree/master/fandango

predict.ECC  

Classify new samples using an Ensemble of Classifier Chains

Description

Uses a trained ECC and Gibbs sampling to predict labels for new samples. .f must return a matrix of probabilities, one row for each observation in newdata.

Usage

# S3 method for class 'ECC'
predict(object, newdata, n.iters = 300, burn.in = 100,
    thin = 2, run_parallel = FALSE, silent = TRUE, .f = NULL, ...)
**predict.ECC**

**Arguments**

- **object**: An object of type ECC returned by `ecc()`.
- **newdata**: A data frame or matrix of features. Must be the same form as the one used with `ecc()`.
- **n.iters**: Number of iterations of the Gibbs sampler.
- **burn.in**: Number of iterations for adaptation (burn-in).
- **thin**: Thinning interval.
- **run.parallel**: Logical flag for utilizing multicore capabilities of the system.
- **silent**: Logical flag indicating whether to have a progress bar (if the 'progress' package is installed) or print progress messages to console.
- **.f**: User-supplied prediction function that corresponds to the type of classifier that was trained in the `ecc()` step. See Details.
- **...**: additional arguments to pass to `.f`.

**Details**

Getting the prediction function correct is very important here. Since this package is a wrapper that can use any classification algorithm as its base classifier, certain assumptions have been made. We assume that the prediction function can return a data.frame or matrix of probabilities with two columns: "0" and "1" because `ecc()` trains on a factor of "0"s and "1"s for more universal consistency.

**Value**

An object of class PUGS containing:

- **y_labels**: inherited from object
- **preds**: A burnt-in, thinned multi-dimensional array of predictions.

**Examples**

```r
x <- movies_train[, -(1:3)]
y <- movies_train[, 1:3]
model_glm <- ecc(x, y, m = 1, .f = glm.fit, family = binomial(link = "logit"))
predictions_glm <- predict(model_glm, movies_test[, -(1:3)],
                         .f = function(glm_fit, newdata) {
                           # Credit for writing the prediction function that works
                           # with objects created through glm.fit goes to Thomas Lumley
                           eta <- as.matrix(newdata) %*% glm_fit$coef
                           output <- glm_fit$family$linkinv(eta)
                           colnames(output) <- "1"
                           return(output)
                         }, n.iters = 10, burn.in = 0, thin = 1)
```

## Not run:

```r
model_c50 <- ecc(x, y, .f = C50::C5.0)
predictions_c50 <- predict(model_c50, movies_test[, -(1:3)],
                           n.iters = 10, burn.in = 0, thin = 1,
                           .f = C50::predict.C5.0, type = "prob")

model_rf <- ecc(x, y, .f = randomForest::randomForest)
predictions_rf <- predict(model_rf, movies_test[, -(1:3)],
                           n.iters = 1000, burn.in = 100, thin = 10,
                           .f = function(rF, newdata) {
                              randomForest::predict.randomForest(rF, newdata, type = "prob")
                           })
```

## End(Not run)

---

### summary.PUGS

#### Gather samples of predictions

**Description**

Collapses the multi-label predictions across sets of classifier chains and iterations into a single set of predictions, either binary or probabilistic.

**Usage**

```r
## S3 method for class 'PUGS'
summary(object, ...)
```

**Arguments**

- `object` A pugs object generated by `predict.ECC`
- `...` type = "prob" for probabilistic predictions, type = "class" for binary (0/1) predictions

**Value**

A matrix of predictions.

**Examples**

```r
x <- movies_train[, -(1:3)]
y <- movies_train[, 1:3]
model_glm <- ecc(x, y, m = 1, .f = glm.fit, family = binomial(link = "logit"))
predictions_glm <- predict(model_glm, movies_test[, -(1:3)],
                          .f = function(glm_fit, newdata) {
```

# Credit for writing the prediction function that works
# with objects created through glm.fit goes to Thomas Lumley

etta <- as.matrix(newdata) %*% glm_fit$coeff
output <- glm_fit$family$linkinv(eta)
colnames(output) <- "1"
return(output)

}, n.iter = 10, burn.in = 0, thin = 1)

summary(predictions_glm, movies_test[, 1:3])

## Not run:

model_c50 <- ecc(x, y, .f = C50::C5.0)
predictions_c50 <- predict(model_c50, movies_test[, -(1:3)],
.n.iter = 10, burn.in = 0, thin = 1,
.f = C50::predict.C5.0, type = "prob")
summary(predictions_c50, movies_test[, 1:3])

model_rf <- ecc(x, y, .f = randomForest::randomForest)
predictions_rf <- predict(model_rf, movies_test[, -(1:3)],
.n.iter = 10, burn.in = 0, thin = 1,
.f = function(rF, newdata){
  randomForest::predict.randomForest(rF, newdata, type = "prob")
})
summary(predictions_rf, movies_test[, 1:3])

## End(Not run)

---

### validate_pugs

**Assess multi-label prediction accuracy**

**Description**

Computes a variety of accuracy metrics for multi-label predictions.

**Usage**

`validate_pugs(object, y)`

**Arguments**

- `object` A PUGS object generated by `predict.ECC`
- `y` A matrix of the same form as the one used with `ecc`

**Value**

A variety of multi-label classification accuracy measurements.
Examples

```r
x <- movies_train[, -(1:3)]
y <- movies_train[, 1:3]

model_glm <- ecc(x, y, m = 1, .f = glm.fit, family = binomial(link = "logit"))
predictions_glm <- predict(model_glm, movies_test[, -(1:3)],
                         .f = function(glm_fit, newdata) {
                           # Credit for writing the prediction function that works
                           # with objects created through glm.fit goes to Thomas Lumley
                           eta <- as.matrix(newdata) %*% glm_fit$coef
                           output <- glm_fit$family$linkinv(eta)
                           colnames(output) <- "1"
                           return(output)
})

validate_pugs(predictions_glm, movies_test[, 1:3])

# Not run:
model_c50 <- ecc(x, y, .f = C50::C5.0)
predictions_c50 <- predict(model_c50, movies_test[, -(1:3)],
                           n.iters = 10, burn.in = 0, thin = 1,
                           .f = C50::predict.C5.0, type = "prob")
validate_pugs(predictions_c50, movies_test[, 1:3])

model_rf <- ecc(x, y, .f = randomForest::randomForest)
predictions_rf <- predict(model_rf, movies_test[, -(1:3)],
                           n.iters = 10, burn.in = 0, thin = 1,
                           .f = function(rF, newdata) {
                             randomForest::predict.randomForest(rF, newdata, type = "prob")
                           })
validate_pugs(predictions_rf, movies_test[, 1:3])

# End(Not run)
```
Index

*Topic datasets
  movies, 4

ecc, 2, 5, 7

MLPUGS (MLPUGS-package), 2
MLPUGS-package, 2
movies, 4
movies_test (movies), 4
movies_train (movies), 4

predict.ECC, 4, 6, 7

summary.PUGS, 6

validate_pugs, 7