Package ‘AdaSampling’

June 27, 2018

Type Package

Title Adaptive Sampling for Positive Unlabeled and Label Noise Learning

Version 1.1

Author Pengyi Yang & Dinuka Perera

Maintainer Pengyi Yang <yangpy7@gmail.com>

Description Implements the adaptive sampling procedure, a framework for both positive unlabeled learning and learning with class label noise. Yang, P., Ormerod, J., Liu, W., Ma, C., Zomaya, A., Yang, J. (2018) <doi:10.1109/TCYB.2018.2816984>.

License GPL-3

Encoding UTF-8

Depends R (>= 3.4.0)

LazyData true

Imports caret (>= 6.0-78) , class (>= 7.3-14), e1071 (>= 1.6-8), stats, MASS

BugReports https://github.com/PengyiYang/AdaSampling/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/PengyiYang/AdaSampling/

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-06-27 15:45:11 UTC

R topics documented:

`adaSample` *Implementation of AdaSampling for positive unlabelled and label noise learning.*

Description

`adaSample()` applies the AdaSampling procedure to reduce noise in the training set, and subsequently trains a classifier from the new training set. For each row (observation) in the test set, it returns the probabilities of it being a positive ("P) or negative ("N") instance, as a two column data frame.

Usage

```
adaSample(Ps, Ns, train.mat, test.mat, classifier = "svm", s = 1, C = 1,
  sampleFactor = 1)
```

Arguments

`Ps`	names (name as index) of positive examples
`Ns`	names (name as index) of negative examples
`train.mat`	training data matrix, without class labels.
`test.mat`	test data matrix, without class labels.
`classifier`	classification algorithm to be used for learning. Current options are support vector machine, `"svm"`, k-nearest neighbour, `"knn"`, logistic regression `"logit"`, or linear discriminant analysis `"lda"`.
`s`	sets the seed.
`C`	sets how many times to run the classifier, C>1 induces an ensemble learning model.
`sampleFactor`	provides a control on the sample size for resampling.

Details

`adaSample()` is an adaptive sampling-based noise reduction method to deal with noisy class labelled data, which acts as a wrapper for traditional classifiers, such as support vector machines, k-nearest neighbours, logistic regression, and linear discriminant analysis.

This process is used to build up a noise-minimized training set that is derived by iteratively resampling the training set, (`train`) based on probabilities derived after its classification.

This sampled training set is then used to train a classifier, which is then executed on the test set. `adaSample()` returns a series of predictions for each row of the test set.

Note that this function does not evaluate the quality of the model and thus does not compare its output to true values of the test set. To assess please see `adaSvmBenchmark()`.

Value

a two column matrix providing classification probabilities of each sample with respect to positive and negative classes

References

Yang, P., Liu, W., Yang. J. (2017) Positive unlabeled learning via wrapper-based adaptive sampling. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 3272-3279

Yang, P., Ormerod, J., Liu, W., Ma, C., Zomaya, A., Yang, J.(2018) AdaSampling for positive-unlabeled and label noise learning with bioinformatics applications. *IEEE Transactions on Cybernetics*, doi:10.1109/TCYB.2018.2816984

Examples

```
# Load the example dataset
data(brca)
head(brca)

# First, clean up the dataset to transform into the required format.
brca.mat <- apply(X = brca[,-10], MARGIN = 2, FUN = as.numeric)
brca.cls <- sapply(X = brca$cla, FUN = function(x) {ifelse(x == ”malignant”, 1, 0)})
rownames(brca.mat) <- paste(”p”, 1:nrow(brca.mat), sep=”_”)

# Introduce 40% noise to positive class and 30% noise to the negative class
set.seed(1)
pos <- which(brca.cls == 1)
neg <- which(brca.cls == 0)
brca.cls.noisy <- brca.cls
brca.cls.noisy[sample(pos, floor(length(pos) * 0.4))] <- 0
brca.cls.noisy[sample(neg, floor(length(neg) * 0.3))] <- 1

# Identify positive and negative examples from the noisy dataset
Ps <- rownames(brca.mat)[which(brca.cls.noisy == 1)]
Ns <- rownames(brca.mat)[which(brca.cls.noisy == 0)]

# Apply AdaSampling method on the noisy data
brca.preds <- adaSample(Ps, Ns, train.mat=brca.mat, test.mat=brca.mat, classifier = ”knn”)
head(brca.preds)

# Orignal accuracy from the labels
accuracy <- sum(brca.cls.noisy == brca.cls) / length(brca.cls)
accuracy

# Accuracy after applying AdaSampling method
accuracyWithAdaSample <- sum(ifelse(brca.preds[,”P”] > 0.5, 1, 0) == brca.cls) / length(brca.cls)
accuracyWithAdaSample
```

adaSvmBenchmark *Benchmarking AdaSampling efficacy on noisy labelled data.*

Description

adaSvmBenchmark() allows a comparison between the performance of an AdaSampling-enhanced SVM (support vector machine)- classifier against the SVM-classifier on its own. It requires a matrix of features (extracted from a labelled dataset), and two vectors of true labels and labels with noise added as desired. It runs an SVM classifier and returns a matrix which displays the specificity (Sp), sensitivity (Se) and F1 score for each of four conditions: "Original" (classifying with true labels), "Baseline" (classifying with noisy labels), "AdaSingle" (classifying using AdaSampling) and "AdaEnsemble" (classifying using AdaSampling in conjunction with an ensemble of models).

Usage

```
adaSvmBenchmark(data.mat, data.cls, data.cls.truth, cvSeed, C = 50,
  sampleFactor = 1)
```

Arguments

data.mat	a rectangular matrix or data frame that can be coerced to a matrix, containing the features of the dataset, without class labels. Rownames (possibly containing unique identifiers) will be ignored.
data.cls	a numeric vector containing class labels for the dataset with added noise. Must be in the same order and of the same length as data.mat. Labels must be 1 for positive observations, and 0 for negative observations.
data.cls.truth	a numeric vector of true class labels for the dataset. Must be the same order and of the same length as data.mat. Labels must be 1, for positive observations, and 0 for negative observations.
cvSeed	sets the seed for cross-validation.
C	sets how many times to run the classifier, for the AdaEnsemble condition. See Description above.
sampleFactor	provides a control on the sample size for resampling.

Details

AdaSampling is an adaptive sampling-based noise reduction method to deal with noisy class labelled data, which acts as a wrapper for traditional classifiers, such as support vector machines, k-nearest neighbours, logistic regression, and linear discriminant analysis. For more details see ?adaSample().

This function runs evaluates the AdaSampling procedure by adding noise to a labelled dataset, and then running support vector machines on the original and the noisy dataset. Note that this function is for benchmarking AdaSampling performance using what is assumed to be a well-labelled dataset. In order to run AdaSampling on a noisy dataset, please see adaSample().

Value

performance matrix

References

Yang, P., Liu, W., Yang. J. (2017) Positive unlabeled learning via wrapper-based adaptive sampling. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 3272-3279

Yang, P., Ormerod, J., Liu, W., Ma, C., Zomaya, A., Yang, J.(2018) AdaSampling for positive-unlabeled and label noise learning with bioinformatics applications. *IEEE Transactions on Cybernetics*, doi:10.1109/TCYB.2018.2816984

Examples

```
# Load the example dataset
data(brca)
head(brca)

# First, clean up the dataset to transform into the required format.
brca.mat <- apply(X = brca[,-10], MARGIN = 2, FUN = as.numeric)
brca.cls <- sapply(X = brca$cla, FUN = function(x) {ifelse(x == "malignant", 1, 0)})
rownames(brca.mat) <- paste("p", 1:nrow(brca.mat), sep="_")

# Introduce 40% noise to positive class and 30% noise to the negative class
set.seed(1)
pos <- which(brca.cls == 1)
neg <- which(brca.cls == 0)
brca.cls.noisy <- brca.cls
brca.cls.noisy[sample(pos, floor(length(pos) * 0.4))] <- 0
brca.cls.noisy[sample(neg, floor(length(neg) * 0.3))] <- 1

# benchmark classification performance with different approaches

adaSvmBenchmark(data.mat = brca.mat, data.cls = brca.cls.noisy, data.cls.truth = brca.cls, cvSeed=1)
```

brca *Wisconsin Breast Cancer Database (1991)*

Description

A cleaned version of the original Wisconsin Breast Cancer dataset containing histological information about 683 breast cancer samples collected from patients at the University of Wisconsin Hospitals, Madison by Dr. William H. Wolberg between January 1989 and November 1991.

Usage

```
brca
```

Format

A data frame with 683 rows and 10 variables:

clt Clump thickness, 1 - 10

ucs Uniformity of cell size, 1 - 10

uch Uniformity of cell shape, 1 - 10

mad Marginal adhesion, 1 - 10

ecs Single epithelial cell size, 1 - 10

nuc Bare nuclei, 1 - 10

chr Bland chromatin, 1 - 10

ncl Normal nucleoli, 1 - 10

mit Mitoses, 1 - 10

cla Class, benign or malignant

Source

`https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/breast-cancer-wisconsin.data`

References

O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", *SIAM News*, Volume 23, Number 5, September 1990, pp 1 & 18.

`singleIter` `singleIter()` *applies a single iteraction of AdaSampling procedure. It returns the probabilities of all samples as being a positive (P) or negative (N) instance, as a two column data frame.*

Description

Classification algorithms included are support vector machines (svm), k-nearest neighbours (knn), logistic regression (logit), and linear discriminant analysis (lda).

Usage

```
singleIter(Ps, Ns, dat, test = NULL, pos.probs = NULL, una.probs = NULL,
  classifier = "svm", sampleFactor, seed)
```

Arguments

`Ps` names (name as index) of positive examples

`Ns` names (name as index) of negative examples

`dat` training data matrix, without class labels.

`test` test data matrix, without class labels. Training data matrix will be used for testing if this is NULL (default).

`pos.probs` a numeric vector of containing probability of positive examples been positive

`una.probs` a numeric vector of containing probability of negative or unannotated examples been negative

`classifier` classification algorithm to be used for learning. Current options are support vector machine, `"svm"`, k-nearest neighbour, `"knn"`, logistic regression `"logit"`, or linear discriminant analysis `"lda"`.

`sampleFactor` provides a control on the sample size for resampling.

`seed` sets the seed.

References

Yang, P., Liu, W., Yang. J. (2017) Positive unlabeled learning via wrapper-based adaptive sampling. *International Joint Conferences on Artificial Intelligence (IJCAI)*, 3272-3279

Yang, P., Ormerod, J., Liu, W., Ma, C., Zomaya, A., Yang, J.(2018) AdaSampling for positive-unlabeled and label noise learning with bioinformatics applications. *IEEE Transactions on Cybernetics*, doi:10.1109/TCYB.2018.2816984

Index