Package ‘AdaSampling’

May 21, 2019

Type Package
Title Adaptive Sampling for Positive Unlabeled and Label Noise Learning
Version 1.3
Author Pengyi Yang
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 6.1.1
NeedsCompilation no
Repository CRAN
Date/Publication 2019-05-21 08:20:04 UTC

R topics documented:

adaSample ..................................................... 2
adaSvmBenchmark ............................................... 4
brca .......................................................... 5
singleIter ....................................................... 6
weightedKNN ................................................... 7
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, weights = NULL)

Arguments

Ps names (each instance in the data has to be named) of positive examples
Ns names (each instance in the data has to be named) 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", linear discriminant analysis "lda", and feature weighted knn, "wkNN".
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.
weights feature weights, required when using weighted knn.

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


Examples

```r
# 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)

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

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

**Description**


**Usage**

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

**Arguments**

- **data.mat**: 이차원 행렬 또는 데이터프레임을 포함하는 데이터셋의 특성에 대한 특성 행렬. 클래스 라벨이 없어. 행 이름(가능한 유니크 아이디)은 무시됩니다.
- **data.cls**: 노이즈가 있는 데이터셋의 수치 벡터인 라벨. 반드시 data.mat와 동일한 순서와 동일한 길이를 가집니다. 1로 설정하면 긍정적 관측을 나타내고, 0으로 설정하면 부정적 관측을 나타냅니다.
- **data.cls.truth**: 데이터셋의 정확한 클래스 라벨의 수치 벡터. 반드시 data.mat와 동일한 순서와 동일한 길이를 가집니다. 1로 설정하면 긍정적 관측을 나타내고, 0으로 설정하면 부정적 관측을 나타냅니다.
- **cvSeed**: 크로스 검증을 위한 세드를 설정합니다.
- **C**: AdaEnsemble 조건에서 classifier를 실행할 횟수를 결정합니다. See Description above.
- **sampleFactor**: resampling을 위한 최대 제약을 제공합니다.

**Details**

AdaSampling은 노이즈가 있는 클래스 라벨화 데이터에 대한 적응성 샘플링 기반의 노이즈 감소 방법으로, 전통적인 분류기, 즉 SVM, k-최고近傍, 로지스틱 회귀, 선형 판별 분석을 포함합니다. More details see ?adaSample().

이 함수는 AdaSampling 프로세스를 실행하여 노이즈에 있는 라벨 데이터를 추가하고 SVM 분류기를 사용하여 원래의 노이즈 데이터를 추가하며 실행합니다. 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


Examples

```r
# 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$cls, 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


References


---

**singleIter**

singleIter() applies a single iteration 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), linear discriminant analysis (lda), feature weighted knn (wKNN).

Usage

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

**Arguments**

- **P**: 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", linear discriminant analysis "lda", and feature weighted knn, "wknn".
- **sampleFactor**: provides a control on the sample size for resampling.
- **seed**: sets the seed.
- **weights**: feature weights, required when using weighted knn.

**References**


**weightedKNN**

*Implementation of a feature weighted k-nearest neighbour classifier.*

**Description**

Implementation of a feature weighted k-nearest neighbour classifier.

**Usage**

`weightedKNN(train.mat, test.mat, cl, k = 3, weights)`

**Arguments**

- **train.mat**: training data matrix, without class labels.
- **test.mat**: test data matrix, without class labels.
- **cl**: class labels for training data.
- **k**: number of nearest neighbour to be used.
- **weights**: weights to be assigned to each feature.
Index

*Topic datasets
   brca, 5

adaSample, 2
adaSvmBenchmark, 4

brca, 5
singleIter, 6
weightedKNN, 7