Package: ebmc (via r-universe)

August 25, 2024

Type Package

Title Ensemble-Based Methods for Class Imbalance Problem

Version 1.0.1

Author Hsiang Hao, Chen

Maintainer ``Hsiang Hao, Chen" <kbman1101@gmail.com>

Description Four ensemble-based methods (SMOTEBoost, RUSBoost, UnderBagging, and SMOTEBagging) for class imbalance problem are implemented for binary classification. Such methods adopt ensemble methods and data re-sampling techniques to improve model performance in presence of class imbalance problem. One special feature offers the possibility to choose multiple supervised learning algorithms to build weak learners within ensemble models. References: Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer (2003) <doi:10.1007/978-3-540-39804-2_12>, Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano (2010) <doi:10.1109/TSMCA.2009.2029559>, R. Barandela, J. S. Sanchez, R. M. Valdovinos (2003) <doi:10.1007/s10044-003-0192-z>, Shuo Wang and Xin Yao (2009) <doi:10.1109/CIDM.2009.4938667>, Yoav Freund and Robert E. Schapire (1997) <doi:10.1006/jcss.1997.1504>.

Depends methods

Imports e1071, rpart, C50, randomForest, pROC, smotefamily

License GPL (>= 3)

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2022-01-10 18:12:44 UTC

Contents

`adam2` *Implementation of AdaBoost.M2*

Description

The function implements AdaBoost.M2 for binary classification. It returns a list of weak learners that are built on random under-sampled training-sets, and a vector of error estimations of each weak learner. The weak learners altogether consist the ensemble model.

Usage

```
adam2(formula, data, size, alg, rf.ntree = 50, svm.ker = "radial")
```

Arguments

`formula`	A formula specify predictors and target variable. Target variable should be a factor of 0 and 1. Predictors can be either numerical and categorical.
`data`	A data frame used for training the model, i.e. training set.
`size`	Ensemble size, i.e. number of weak learners in the ensemble model.
`alg`	The learning algorithm used to train weak learners in the ensemble model. *cart*, *c50*, *rf*, *nb*, and *svm* are available. Please see Details for more information.
`rf.ntree`	Number of decision trees in each forest of the ensemble model when using *rf* (Random Forest) as base learner. Integer is required.
`svm.ker`	Specifying kernel function when using svm as base algorithm. Four options are available: **linear**, **polynomial**, **radial**, and **sigmoid**. Default is radial. Equivalent to that in e1071::svm().

Details

AdaBoost.M2 is an extension of AdaBoost. AdaBoost.M2 introduces *pseudo-loss*, which is a more sophisticated method to estimate error and update instance weight in each iteration compared to AdaBoost and AdaBoost.M1. Although AdaBoost.M2 is originally implemented with decision tree, this function makes it possible to use other learning algorithms for building weak learners.

Argument *alg* specifies the learning algorithm used to train weak learners within the ensemble model. Totally five algorithms are implemented: **cart** (Classification and Regression Tree), **c50** (C5.0 Decision Tree), **rf** (Random Forest), **nb** (Naive Bayes), and **svm** (Support Vector Machine).

When using Random Forest as base learner, the ensemble model is consisted of forests and each forest contains a number of trees.

The function requires the target varible to be a factor of 0 and 1, where 1 indicates minority while 0 indicates majority instances. Only binary classification is implemented in this version.

The object class of returned list is defined as *modelBst*, which can be directly passed to predict() for predicting test instances.

Value

The function returns a list containing two elements:

`weakLearners` A list of weak learners.

`errorEstimation`
Error estimation of each weak learner. Calculated by using (pseudo_loss + smooth) / (1 - pseudo_loss + smooth). *smooth* helps prevent error rate = 0 resulted from perfect classfication during trainging iterations. For more information, please see Schapire et al. (1999) Section 4.2.

References

Freund, Y. and Schapire, R. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences. 55, pp. 119-139.

Freund, Y. and Schapire, R. 1996. Experiments with a new boosting algorithm. Machine Learning: In Proceedings of the 13th International Conference. pp. 148-156

Schapire, R. and Singer, Y. 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning. 37(3). pp. 297-336.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 42(4), pp. 463-484.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))
model1 <- adam2(Species ~ ., data = iris, size = 10, alg = "c50")
model2 <- adam2(Species ~ ., data = iris, size = 20, alg = "rf", rf.ntree = 100)
model3 <- adam2(Species ~ ., data = iris, size = 40, alg = "svm", svm.ker = "sigmoid")
```

---

measure *Calculating Performance Measurement in Class Imbalance Problem*

Description

The function is an interation of multiple performance measurements that can be used to assess model performance in class imbalance problem. Totally six measurements are included.

Usage

```
measure(label, probability, metric, threshold = 0.5)
```

Arguments

label	A vector of actual labels of target variable in test set.
probability	A vector of probability estimated by the model.
metric	Measurement used for assessing model performance. **auc**, **gmean**, **tpr**, **tnr**, **f**, and **acc** are available. Please see Details for more information.
threshold	Probability threshold for determining the class of instances. A numerical value ranging from 0 to 1. Default is 0.5

Details

This function integrates six common measurements. It uses pROC::roc() and pROC::auc() to calculate **auc** (Area Under Curve), while calculates other measurements without dependency on other package: **gmean** (Geometric Mean), **tpr** (True Positive Rate), **tnr** (True Negative Rate),and **f** (F-Measure).

acc (Accuracy) is also included for any possible use, although such measurement can be misleading when the classes of test set is highly imbalanced.

threshold is the probability cutoff for determing the predicted class of instances. For AUC, users do not need to specify threshold because AUC is not affected by the probability cutoff. However, the threshold is required for other five measurements.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))

# Creat training and test set
samp <- sample(nrow(iris), nrow(iris) * 0.7)
train <- iris[samp, ]
test <- iris[-samp, ]

# Model building and prediction
model <- rus(Species ~ ., data = train, size = 10, alg = "c50")
prob <- predict(model, newdata = test, type = "prob")

# Calculate measurements
auc <- measure(label = test$Species, probability = prob, metric = "auc")
gmean <- measure(label = test$Species, probability = prob, metric = "gmean", threshold = 0.5)
```

predict.modelBag *Predict Method for modelBag Object*

Description

Predicting instances in test set using modelBag object

Usage

```
 ## S3 method for class 'modelBag'
predict(object, newdata, type = "prob", ...)
```

Arguments

object	A object of *modelBag* class.
newdata	A *data frame* object containing new instances.
type	Types of output, which can be **prob** (probability) and **class** (predicted label). Default is prob.
...	Not used currently.

Value

Two type of output can be selected:

prob	Estimated probability of being a minority instance (i.e. 1). The probability is averaged by using an equal-weight majority vote by all weak learners.
class	Predicted class of the instance. Instances of probability larger than 0.5 are predicted as 1, otherwise 0.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))
samp <- sample(nrow(iris), nrow(iris) * 0.7)
train <- iris[samp, ]
test <- iris[-samp, ]
model <- ub(Species ~ ., data = train, size = 10, alg = "c50") # Build UnderBagging model
prob <- predict(model, newdata = test, type = "prob") # return probability estimation
pred <- predict(model, newdata = test, type = "class") # return predicted class
```

`predict.modelBst` *Predict Method for modelBst Object*

Description

Predicting instances in test set using modelBst object

Usage

```
  ## S3 method for class 'modelBst'
predict(object, newdata, type = "prob", ...)
```

Arguments

`object`	A object of *modelBst* class.
`newdata`	A *data frame* object containing new instances.
`type`	Types of output, which can be **prob** (probability) and **class** (predicted label). Default is prob.
`...`	Not used currently.

Value

Two type of output can be selected:

`prob`	Estimated probability of being a minority instance (i.e. 1). The probability is averaged by using a majority vote by all weak learners, weighted by error estimation.
`class`	Predicted class of the instance. Instances of probability larger than 0.5 are predicted as 1, otherwise 0.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))
samp <- sample(nrow(iris), nrow(iris) * 0.7)
train <- iris[samp, ]
test <- iris[-samp, ]
model <- rus(Species ~ ., data = train, size = 10, alg = "c50") # Build RUSBoost model
prob <- predict(model, newdata = test, type = "prob") # return probability estimation
pred <- predict(model, newdata = test, type = "class") # return predicted class
```

rus *Implementation of RUSBoost*

Description

The function implements RUSBoost for binary classification. It returns a list of weak learners that are built on random under-sampled training-sets, and a vector of error estimations of each weak learner. The weak learners altogether consist the ensemble model.

Usage

```
rus(formula, data, size, alg, ir = 1, rf.ntree = 50, svm.ker = "radial")
```

Arguments

formula	A formula specify predictors and target variable. Target variable should be a factor of 0 and 1. Predictors can be either numerical and categorical.
data	A data frame used for training the model, i.e. training set.
size	Ensemble size, i.e. number of weak learners in the ensemble model.
alg	The learning algorithm used to train weak learners in the ensemble model. *cart*, *c50*, *rf*, *nb*, and *svm* are available. Please see Details for more information.
ir	Imbalance ratio. Specifying how many times the under-sampled majority instances are over minority instances. Interger is not required and so such as ir = 1.5 is allowed.
rf.ntree	Number of decision trees in each forest of the ensemble model when using *rf* (Random Forest) as base learner. Integer is required.
svm.ker	Specifying kernel function when using svm as base algorithm. Four options are available: **linear**, **polynomial**, **radial**, and **sigmoid**. Default is radial. Equivalent to that in e1071::svm().

Details

Based on AdaBoost.M2, RUSBoost uses random under-sampling to reduce majority instances in each iteration of training weak learners. A 1:1 under-sampling ratio (i.e. equal numbers of majority and minority instances) is set as default.

The function requires the target varible to be a factor of 0 and 1, where 1 indicates minority while 0 indicates majority instances. Only binary classification is implemented in this version.

Argument *alg* specifies the learning algorithm used to train weak learners within the ensemble model. Totally five algorithms are implemented: **cart** (Classification and Regression Tree), **c50** (C5.0 Decision Tree), **rf** (Random Forest), **nb** (Naive Bayes), and **svm** (Support Vector Machine). When using Random Forest as base learner, the ensemble model is consisted of forests and each forest contains a number of trees.

ir refers to the intended imbalance ratio of training sets for manipulation. With ir = 1 (default), the numbers of majority and minority instances are equal after class rebalancing. With ir = 2, the

number of majority instances is twice of that of minority instances. Interger is not required and so such as ir = 1.5 is allowed.

The object class of returned list is defined as *modelBst*, which can be directly passed to predict() for predicting test instances.

Value

The function returns a list containing two elements:

`weakLearners` A list of weak learners.

`errorEstimation`
Error estimation of each weak learner. Calculated by using (pseudo_loss + smooth) / (1 - pseudo_loss + smooth). *smooth* helps prevent error rate = 0 resulted from perfect classfication during trainging iterations. For more information, please see Schapire et al. (1999) Section 4.2.

References

Seiffert, C., Khoshgoftaar, T., Hulse, J., and Napolitano, A. 2010. RUSBoost: A Hybrid Approach to Alleviating Class Imbalance. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans. 40(1), pp. 185-197.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 42(4), pp. 463-484.

Freund, Y. and Schapire, R. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences. 55, pp. 119-139.

Freund, Y. and Schapire, R. 1996. Experiments with a new boosting algorithm. Machine Learning: In Proceedings of the 13th International Conference. pp. 148-156

Schapire, R. and Singer, Y. 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning. 37(3). pp. 297-336.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))
model1 <- rus(Species ~ ., data = iris, size = 10, alg = "c50", ir = 1)
model2 <- rus(Species ~ ., data = iris, size = 20, alg = "rf", ir = 1, rf.ntree = 100)
model3 <- rus(Species ~ ., data = iris, size = 40, alg = "svm", ir = 1, svm.ker = "sigmoid")
```

sbag *Implementation of SMOTEBagging*

Description

The function implements SMOTEBagging for binary classification. It returns a list of weak learners that are built on training-sets manipulated by SMOTE and random over-sampling. They together consist the ensemble model.

Usage

```
sbag(formula, data, size, alg, smote.k = 5, rf.ntree = 50, svm.ker = "radial")
```

Arguments

formula	A formula specify predictors and target variable. Target variable should be a factor of 0 and 1. Predictors can be either numerical and categorical.
data	A data frame used for training the model, i.e. training set.
size	Ensemble size, i.e. number of weak learners in the ensemble model.
alg	The learning algorithm used to train weak learners in the ensemble model. *cart*, *c50*, *rf*, *nb*, and *svm* are available. Please see Details for more information.
smote.k	Number of k applied in SMOTE algorithm. Default is 5.
rf.ntree	Number of decision trees in each forest of the ensemble model when using *rf* (Random Forest) as base learner. Integer is required.
svm.ker	Specifying kernel function when using svm as base algorithm. Four options are available: **linear**, **polynomial**, **radial**, and **sigmoid**. Default is radial. Equivalent to that in e1071::svm().

Details

SMOTEBagging uses both SMOTE (Synthetic Minority Over-sampling TEchnique) and random over-sampling to increase minority instances in each bag of Bagging in order to rebalance class distribution. The manipulated training sets contain equal numbers of majority and minority instances, but the proportions of minority instances from SMOTE and random over-sampling vary for different bags, determined by an assigned re-sampling rate *a*. The re-sampling rate *a* is always the multiple of 10, and the function automatically generates a vector of *a*, therefore users do not need to self-define.

The function requires the target varible to be a factor of 0 and 1, where 1 indicates minority while 0 indicates majority instances. Only binary classification is implemented in this version.

Argument *alg* specifies the learning algorithm used to train weak learners within the ensemble model. Totally five algorithms are implemented: **cart** (Classification and Regression Tree), **c50** (C5.0 Decision Tree), **rf** (Random Forest), **nb** (Naive Bayes), and **svm** (Support Vector Machine). When using Random Forest as base learner, the ensemble model is consisted of forests and each forest contains a number of trees.

The object class of returned list is defined as *modelBag*, which can be directly passed to predict() for predicting test instances.

References

Wang, S. and Yao, X. 2009. Diversity Analysis on Imbalanced Data Sets by Using Ensemble Models. IEEE Symposium on Computational Intelligence and Data Mining, CIDM '09.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 42(4), pp. 463-484.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))
model1 <- sbag(Species ~ ., data = iris, size = 10, alg = "c50")
model2 <- sbag(Species ~ ., data = iris, size = 20, alg = "rf", rf.ntree = 100)
model3 <- sbag(Species ~ ., data = iris, size = 40, alg = "svm", svm.ker = "sigmoid")
```

---

sbo *Implementation of SMOTEBoost*

Description

The function implements SMOTEBoost for binary classification. It returns a list of weak learners that are built on SMOTE-manipulated training-sets, and a vector of error estimations of each weak learner. The weak learners altogether consist the ensemble model.

Usage

```
sbo(formula, data, size, alg, over = 100, smote.k = 5, rf.ntree = 50, svm.ker = "radial")
```

Arguments

`formula`	A formula specify predictors and target variable. Target variable should be a factor of 0 and 1. Predictors can be either numerical and categorical.
`data`	A data frame used for training the model, i.e. training set.
`size`	Ensemble size, i.e. number of weak learners in the ensemble model.
`alg`	The learning algorithm used to train weak learners in the ensemble model. *cart*, *c50*, *rf*, *nb*, and *svm* are available. Please see Details for more information.
`over`	Specifying over-sampling rate of SMOTE. Only multiple of 100 is acceptable.
`smote.k`	Number of k applied in SMOTE algorithm. Default is 5.
`rf.ntree`	Number of decision trees in each forest of the ensemble model when using *rf* (Random Forest) as base learner. Integer is required.
`svm.ker`	Specifying kernel function when using svm as base algorithm. Four options are available: **linear**, **polynomial**, **radial**, and **sigmoid**. Default is radial. Equivalent to that in e1071::svm().

Details

Based on AdaBoost.M2, SMOTEBoost uses SMOTE (Synthetic Minority Over-sampling TEchnique) to increase minority instances in each iteration of training weak learners. An over-sampling rate of SMOTE can be defined by users with argument *over*.

The function requires the target varible to be a factor of 0 and 1, where 1 indicates minority while 0 indicates majority instances. Only binary classification is implemented in this version.

Argument *alg* specifies the learning algorithm used to train weak learners within the ensemble model. Totally five algorithms are implemented: **cart** (Classification and Regression Tree), **c50** (C5.0 Decision Tree), **rf** (Random Forest), **nb** (Naive Bayes), and **svm** (Support Vector Machine). When using Random Forest as base learner, the ensemble model is consisted of forests and each forest contains a number of trees.

The object class of returned list is defined as *modelBst*, which can be directly passed to predict() for predicting test instances.

Value

The function returns a list containing two elements:

`weakLearners` A list of weak learners.

`errorEstimation`
Error estimation of each weak learner. Calculated by using (pseudo_loss + smooth) / (1 - pseudo_loss + smooth). *smooth* helps prevent error rate = 0 resulted from perfect classfication during trainging iterations. For more information, please see Schapire et al. (1999) Section 4.2.

References

Chawla, N., Lazarevic, A., Hall, L., and Bowyer, K. 2003. SMOTEBoost: Improving Prediction of the Minority Class in Boosting. In Proceedings European Conference on Principles of Data Mining and Knowledge Discovery. pp. 107-119

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 42(4), pp. 463-484.

Freund, Y. and Schapire, R. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences. 55, pp. 119-139.

Freund, Y. and Schapire, R. 1996. Experiments with a new boosting algorithm. Machine Learning: In Proceedings of the 13th International Conference. pp. 148-156

Schapire, R. and Singer, Y. 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. Machine Learning. 37(3). pp. 297-336.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))
model1 <- sbo(Species ~ ., data = iris, size = 10, over = 100, alg = "c50")
```

```
model2 <- sbo(Species ~ ., data = iris, size = 20, over = 200, alg = "rf", rf.ntree = 100)
model3 <- sbo(Species ~ ., data = iris, size = 40, over = 300, alg = "svm", svm.ker = "sigmoid")
```

---

ub *Implementation of UnderBagging*

Description

The function implements UnderBagging for binary classification. It returns a list of weak learners that are built on random under-sampled training-sets. They together consist the ensemble model.

Usage

```
ub(formula, data, size, alg, ir = 1, rf.ntree = 50, svm.ker = "radial")
```

Arguments

`formula`	A formula specify predictors and target variable. Target variable should be a factor of 0 and 1. Predictors can be either numerical and categorical.
`data`	A data frame used for training the model, i.e. training set.
`size`	Ensemble size, i.e. number of weak learners in the ensemble model.
`alg`	The learning algorithm used to train weak learners in the ensemble model. *cart*, *c50*, *rf*, *nb*, and *svm* are available. Please see Details for more information.
`ir`	Imbalance ratio. Specifying how many times the under-sampled majority instances are over minority instances. Interger is not required and so such as ir = 1.5 is allowed.
`rf.ntree`	Number of decision trees in each forest of the ensemble model when using *rf* (Random Forest) as base learner. Integer is required.
`svm.ker`	Specifying kernel function when using svm as base algorithm. Four options are available: **linear**, **polynomial**, **radial**, and **sigmoid**. Default is radial. Equivalent to that in e1071::svm().

Details

UnderBagging uses random under-sampling to reduce majority instances in each bag of Bagging in order to rebalance class distribution. A 1:1 under-sampling ratio (i.e. equal numbers of majority and minority instances) is set as default.

The function requires the target varible to be a factor of 0 and 1, where 1 indicates minority while 0 indicates majority instances. Only binary classification is implemented in this version.

Argument *alg* specifies the learning algorithm used to train weak learners within the ensemble model. Totally five algorithms are implemented: **cart** (Classification and Regression Tree), **c50** (C5.0 Decision Tree), **rf** (Random Forest), **nb** (Naive Bayes), and **svm** (Support Vector Machine). When using Random Forest as base learner, the ensemble model is consisted of forests and each forest contains a number of trees.

ir refers to the intended imbalance ratio of training sets for manipulation. With ir = 1 (default), the numbers of majority and minority instances are equal after class rebalancing. With ir = 2, the number of majority instances is twice of that of minority instances. Interger is not required and so such as ir = 1.5 is allowed.

The object class of returned list is defined as *modelBag*, which can be directly passed to predict() for predicting test instances.

References

Barandela, R., Sanchez, J., and Valdovinos, R. 2003. New Applications of Ensembles of Classifiers. Pattern Analysis and Applications. 6(3), pp. 245-256.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., and Herrera, F. 2012. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 42(4), pp. 463-484.

Examples

```
data("iris")
iris <- iris[1:70, ]
iris$Species <- factor(iris$Species, levels = c("setosa", "versicolor"), labels = c("0", "1"))
model1 <- ub(Species ~ ., data = iris, size = 10, alg = "c50", ir = 1)
model2 <- ub(Species ~ ., data = iris, size = 20, alg = "rf", ir = 1, rf.ntree = 100)
model3 <- ub(Species ~ ., data = iris, size = 40, alg = "svm", ir = 1, svm.ker = "sigmoid")
```

Index