Conformal: an R package to calculate prediction errors in the conformal prediction framework

October 21, 2014

`conformal`	*conformal: an R package to calculate prediction errors in the conformal prediction framework*

Description

conformal permits the calculation of prediction errors in the conformal prediction framework: (i) p.values for classification, and (ii) confidence intervals for regression.

Details

Assessing the reliability of individual predictions is foremost in machine learning to determine the applicability domain of a predictive model, be it in the context of classification or regression. The applicability domain is usually defined as the amount (and thus regions) of descriptor space to which a model can be reliably applied. Conformal prediction is an algorithm-independent technique, i.e. it works with any predictive method such as SVM, PLS, etc.., which outputs confidence regions for individual predictions in the case of regression, and p.values for the categories in classification.

Regression

In regression, the confidence level, ϵ, is controlled by the user, and thus the interpretation is straightforward. For instance, it we choose a confidence level of 80% the true value for new datapoints will lie outside the predicted confidence intervals in at most 20% of the cases.

In the conformal prediction framework, the datapoints in the training set are used to define how unlikely a new datapoint is with respect to the data presented to the model in the training phase. The conformity for a given datapoint, x_i, with respect to the training set is quantified with a nonconformity score, α, calculated with a nonconformity measure (e.g. `StandardMeasure`), which here we define as:

$$\alpha_i = \frac{|y_i - \widetilde{y}_i|}{\widetilde{\rho}_i}$$

where α_i is the nonconformity measure, y_i and $\widetilde{y}_i$ are respectively the the observed and the predicted value calculated with an error model, and $\widetilde{\rho}_i$ is the predicted error for x_i calculated with an error model. The nonconformity scores are then translated into a confidence region for a user-defined

confidence level, ϵ.
In order to calculate confidence intervals, we need a point prediction model, to predict the response variable (y), and an error model, to predict errors in prediction ($\widehat{\rho}$). The point prediction and error models can be generated with any machine learning algorithm, hence algorithm-independent (see above). Both the point prediction and error models need to be trained with cross-validation in order to calculate the vector of nonconformity scores for the training set.
The cross-validation predictions generated when training the point prediction model serve to calculate the errors in prediction for the datapoints in the training set. The error model is then generated by training a machine learning model on the training set using these errors as the dependent variable. The (i) cross-validated predictions from the point prediction model, and (ii) the cross-validated errors in prediction from the error model, are used to generate the vector of nonconformity scores for the training set. This vector, after being sorted in increasing order, can be defined as:

$$\alpha_{training} = \{\alpha_{training\ i}\}_{i}^{N_{training}}$$

where $N_{training}$ is the number of datapoints in the training set.

To generate the confidence intervals for an external set, the α value associated to the user-defined confidence level, α_{ϵ}, is calculated as:

$$\alpha_{\epsilon} = \alpha_{training\ i} \;\; if \;\; i \equiv |N_{training} * \epsilon|$$

where $\equiv$ indicates equality. Next, the errors in prediction, $\widetilde{\rho}_{ext}$, and the value for the response variable, y_{ext}, for the datapoints in an external dataset are predicted with the error and the point prediction models, respectively.

Individual confidence intervals for each datapoint in the external set are calculated as

$$|y_{ext} - \widetilde{y}_{ext}| = \alpha_{\epsilon} * \widetilde{\rho}_{ext}$$

Classification

References

Isidro Cortes <isidrolauscher@gmail.com>. conformal: an R package to calculate prediction errors in the conformal prediction framework.

Shafer et al. JMLR, 2008, 9, pp 371-421. http://machinelearning.wustl.edu/mlpapers/paper_files/shafer08a.pdf

Norinder et al. J. Chem. Inf. Model., 2014, 54 (6), pp 1596-1603. DOI: 10.1021/ci5001168 http://pubs.acs.org/doi/abs/10.1021/ci5001168

`ConformalClassification`

Conformal Prediction For Classification

Description

R class to p.values for individual predictions according to the conformal prediction framework.

Usage

```
ConformalClassification(...)
```

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

References

Norinder et al. J. Chem. Inf. Model., 2014, 54 (6), pp 1596-1603 DOI: 10.1021/ci5001168 http://pubs.acs.org/doi/abs/10.1021/ci5001168

See Also

ConformalRegression

Examples

```
# Optional for parallel training
#library(doMC)
#registerDoMC(cores=4)

data(LogS)

# convert data to categorical
LogSTrain[LogSTrain > -4] <- 1
LogSTrain[LogSTrain <= -4] <- 2
LogSTest[LogSTest > -4] <- 1
LogSTest[LogSTest <= -4] <- 2

LogSTrain <- factor(LogSTrain)
LogSTest <- factor(LogSTest)

algorithm <- "rf"

trControl <- trainControl(method = "cv",  number=5,savePredictions=TRUE,
                          predict.all=TRUE,keep.forest=TRUE,norm.votes=TRUE)
set.seed(3)
model <- train(LogSDescsTrain, LogSTrain, algorithm,type="classification",
               trControl=trControl)


# Instantiate the class and get the p.values
example <- ConformalClassification$new()
example$CalculateCVAlphas(model=model)
example$CalculatePValues(new.data=LogSDescsTest)
example$p.values$P.values
example$p.values$Significance_p.values
```

ConformalRegression *Conformal Prediction for Regression*

Description

R class to create and visualize confidence intervals for individual predictions according to the conformal prediction framework.

Details

The class ConformalRegression contains the following fields:

- PointPredictionModel: stores a point prediction model.
- ErrorModel: stores an error model. Error models can be trained with the function `ErrorModel`.
- confidence: stores the level of confidence used to calculate the confidence intervals. This value is defined when instantiating a new class. Values are in the 0-1 range. Interpretation: for instance, a confidence level of 0.8 (80%) means that, at most, 20% of the confidence intervals will not contain the true value.
- data.new: stores the descriptors corresponding to an external set.
- alphas: stores the nonconformity scores for the datapoints used to train the point prediction model (PointPredictionModel) with the method CalculateAlphas.
- errorPredictions: errors in prediction for an external set predicted with an error model (stored in the field ErrorModel). These errors are generated when calling the method GetConfidenceIntervals. These errors serve to calculate the nonconformity scores (alphas) for the an external set. In the conformal prediction framework, the confidence intervals are derived from the nonconformity values (see methods CalculateAlphas and GetConfidenceIntervals).
- pointPredictions: point predictions for an external set calculated with the point prediction model (stored in the field PointPredictionModel). These predictions are generated with the method GetConfidenceIntervals.
- intervals: numeric vector with the errors in prediction for the external set (data.new) calculated in the conformal prediction framework (not with an error model). These intervals are calculated when calling the method GetConfidenceIntervals.
- plot: stores a correlation plot for the observed against the predicted values, with individual confidence intervals, for the datapoints in an external set. The plot is a ggplot2 object which can be further customized. The plot is generated with the method CorrelationPlot.

The class ConformalRegression contains the following methods:

- initialize: this method is called when you create an instance of the class. The confidence level (field confidence) needs to be defined.
- CalculateAlphas: this method calculates the vector of nonconformity scores for the datapoints in the traning set. These scores (or alphas) are stored in the field alphas. This method requires a point prediction model (argument model), an error model (argument error_model), and a nonconformity measure (argument ConformityMeasure), such as `StandardMeasure`.
- GetConfidenceIntervals: this methods calculates confidence intervals for individual predictions in the conformal prediction framework. The methods requires an external set for which the confidence intervals will be calculated. The dimensionality of these descriptors must be the same as the one used for the datapoints used to train the point prediction and the error model. The method uses the point prediciton and the error models stored in the fields PointPredictionModel and ErrorModel, respectively. Confidence intervals are calculated according to Norinder et al. 2014. Confidence intervals are defined as: point predictions (stored in the field pointPredictions) +/- the output of the method GetConfidenceIntervals, which is stored in the field intervals.
- CorrelationPlot: this method generates a correlation plot for the observed against the predicted values for an external set, along with individual confidence intervals. The plot is stored in the field plot.

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

References

Norinder et al. J. Chem. Inf. Model., 2014, 54 (6), pp 1596-1603 DOI: 10.1021/ci5001168 http://pubs.acs.org/doi/abs/10.1021/ci5001168

See Also

conformal

Examples

```
##############################################
### Example
##############################################

# Optional for parallel training
#library(doMC)
#registerDoMC(cores=4)

data(LogS)

algorithm <- "svmRadial"
tune.grid <- expand.grid(.sigma = expGrid(power.from=-10, power.to=-6, power.by=1, base=2),
                         .C = expGrid(power.from=4, power.to=10, power.by=2, base=2))
trControl <- trainControl(method = "cv",  number=5,savePredictions=TRUE)
set.seed(3)
model <- train(LogSDescsTrain, LogSTrain, algorithm,
               tuneGrid=tune.grid,
               trControl=trControl)


# Train an error model
error_model <- ErrorModel(PointPredictionModel=model,x.train=,LogSDescsTrain,
                          savePredictions=TRUE,algorithm=algorithm,
                          trControl=trControl, tune.grid=tune.grid)

# Instantiate the class and get the confidence intervals
example <- ConformalRegression$new()
example$CalculateAlphas(model=model,error_model=error_model,ConformityMeasure=StandardMeasure)
example$GetConfidenceIntervals(new.data=LogSDescsTest)
example$CorrelationPlot(obs=LogSTest)
example$plot
```

`ErrorModel` *Calculation of an error model*

Description

This function permits the calculation of an error model from (i) a training set, and (ii) a caret model trained on this set to predict the response variable of interest using cross-validation (point prediction model). The cross-validation predictions are extracted from the point prediction model. The errors

in prediction for the cross-validation predictions are then calculated. These errors which serve as the response variable for the error model (i.e. the error model predicts errors in prediction). The error model uses as descriptors the same descriptors used to train the point prediction model. These descriptors are input to the function ErrorModel through the argument "x.train".

Usage

```
ErrorModel(PointPredictionModel, x.train, algorithm = "svmRadial", ...)
```

Arguments

`PointPredictionModel`	Point prediction model from which the cross-validation predictions will be extracted.
`x.train`	Descriptors for the datapoints in the training set used to train the point prediction model, and which will also serve to train the error model.
`algorithm`	The machine learning algorithm to be used to train the error model. The defaul value is Support Vector Machine with radial kernel ("svmRadial").
`...`	Additional arguments that can be passed to the train function from the R package caret to train the error model.

Value

A list of class train containing the error model (caret model).

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

References

http://caret.r-forge.r-project.org/training.html

Kuhn (2008), "Building Predictive Models in R Using the caret" (http://www.jstatsoft.org/v28/i05/)

See Also

`ConformalRegression` `GetCVPreds`

`expGrid` *Exponential Grid Definition*

Description

The function defines an exponential series, which can be used, e.g. when defining the parameter space when training some models such as Support Vector Machines or Gaussian Processes.

Usage

```
expGrid(power.from, power.to, power.by, base)
```

Arguments

power.from	The starting exponential of the series.
power.to	The latest exponential of the series.
power.by	The exponential step of the series.
base	The base of the exponential series.

Value

A vector with the exponential series.

Author(s)

Isidro Cortes <isidrolauscher@gmail.com>

Examples

```
expGrid(power.from=-10,power.to=10,power.by=2,base=10)
```

GetCVPreds *Extract the cross-validation predictions for the datapoints in the training set*

Description

This function extracts the cross-validation predictions from a caret model trained with cross-validation. If grid-search is used to optimize the hyperparamter values, the function GetCVPreds extracts the cross-validation predictions corresponding to the optimal hyperparameter values.

Usage

```
GetCVPreds(model)
```

Arguments

model	A caret model trained with cross-validation.

Value

A data frame containing the observed and the predicted values for the datapoints in the training set, their index according to the training set, the optimal hyperparameter values, and the fold index.

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

LogS *Small Molecule Solubility (LogS) Data*

Usage

```
data(LogS)
```

Details

This dataset comprises the aqueous solubility (S) values at a temperature of 20-25 $^\circ$C in mol/L, expressed as logS, for 1,708 small molecules reported by Wang et al. Compound structures were standardized with the function StandardiseMolecules from the R package camb using the default parameters: (i) all molecules were kept irrespective of the numbers of fluorines, iodines, chlorine, and bromines present in their strucuture, or (ii) of their molecular mass. 905 one-dimensional topological and physicochemical descriptors were calculated with the function GeneratePadelDescriptors from the R package camb which invokes the PaDEL-Descriptor Java library. Near zero variance and highly-correlated descriptors were removed with the functions (i) RemoveNearZeroVarianceFeatures (cut-off value of 30/1), and (ii) RemoveHighlyCorrelatedFeatures (cut-off value of 0.95) After applying these steps the dataset consists of 1,606 molecules encoded with 211 descriptors.

Using data(LogS) exposes 4 objects:

(i) LogSDescsTrain is a data frame with PaDEL descriptors for the datapoints in the training set (70% of the data). (ii) LogSTrain is a numeric vector containing the data solubility values for the datapoints in the training set. (iii) LogSDescsTest is a data frame with PaDEL descriptors for the datapoints in the test set (30% of the data). (iv) LogSTest is a numeric vector containing the data solubility values for the datapoints in the test set.

References

Wang et al. J. Chem. Inf. Model., 2007, 47 (4), pp 1395-1404 DOI: 10.1021/ci700096r http://pubs.acs.org/doi/abs/10.1021/ci700096r

Examples

```
# To use the data
data(LogS)
```

StandardMeasure *Default Nonconformity measure used in regression (ConformalRegression class)*

Description

Nonconformity measure used by default in the class ConformalRegression to calculate nonconformity scores (alpha). StandardMeasure implements the following nonconformity measure:

alpha = |obs - pred| / pred_error

where alpha is the nonconformity measure, obs is the observed value, pred the predicted value with a point prediction model, and pred_error is the error in prediction predicted with an error model. See conformal for further information about how to derive individual confidence intervals.

Usage

```
StandardMeasure(obs, pred, error)
```

Arguments

obs	Observed values for the response variable.
pred	Predicted values for the response variable.
error	Predicted errors calculated with an error model.

Value

A numeric vector with the nonconformity scores (alpha).

Author(s)

Isidro Cortes-Ciriano <isidrolauscher@gmail.com>

References

Norinder et al. J. Chem. Inf. Model., 2014, 54 (6), pp 1596-1603 DOI: 10.1021/ci5001168 http://pubs.acs.org/doi/abs/10.1021/ci5001168

See Also

ConformalRegression

Index