Package ‘ELMR’

November 28, 2015

Title Extreme Machine Learning (ELM)
Version 1.0
Author Alessio Petrozziello [aut, cre]
Maintainer Alessio Petrozziello <alessio.petrozziello@port.ac.uk>
Description Training and prediction functions are provided for the Extreme Learning Machine algorithm (ELM). The ELM use a Single Hidden Layer Feedforward Neural Network (SLFN) with random generated weights and no gradient-based backpropagation. The training time is very short and the online version allows to update the model using small chunk of the training set at each iteration. The only parameter to tune is the hidden layer size and the learning function.
Depends R (>= 3.2.2)
License GPL-2 | GPL-3
LazyData true
RoxygenNote 5.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2015-11-28 14:53:50

R topics documented:

OSelm_train.formula .......................... 2
OSelm_training ................................ 2
predict_elm ......................... 3
preProcess ......................... 4

Index 5
**OSelm_train**

*Trains an extreme learning machine with random weights*

**Description**

Trains an extreme learning machine with random weights

**Usage**

```r
OSelm_train(formula, data, elm_type, n_hidden_neurons, activation_function, p, Block)
```

**Arguments**

- `formula`: a symbolic description of the model to be fitted.
- `data`: training data frame containing the variables specified in formula.
- `elm_type`: select if the ELM must perform a "regression" or "classification".
- `n_hidden_neurons`: number of neurons in the hidden layer.
- `activation_function`: "rbf" for radial basis function with Gaussian kernels, "sig" for sigmoidal function, "sin" for sine function, "hardlim" for hard limit function.
- `p`: size of the first block to be processed.
- `Block`: size of each chunk to be processed at each step.

**Value**

returns all the parameters used in the function, the weight matrix, the labels for the classification, the number of classes found, the bias, the beta activation function and the accuracy on the training set.

**OSelm_training**

*Trains an online sequential extreme learning machine with random weights*

**Description**

Trains an online sequential extreme learning machine with random weights

**Usage**

```r
OSelm_training(p, y, Elm_Type, n_hidden_neurons, activation_function, N0, Block)
```
**predict_elm**

**Arguments**

- **p**: dataset used to perform the training of the model
- **y**: classes vector for classification or regressors for regression
- **Elm_Type**: select if the ELM must perform a "regression" or "classification"
- **nHiddenNeurons**: number of neurons in the hidden layer
- **ActivationFunction**: "rbf" for radial basis function with Gaussian kernels, "sig" for sigmoidal function, "sin" for sine function, "hardlim" for hard limit function
- **N0**: size of the first block to be processed
- **Block**: size of each chunk to be processed at each step

**Value**

returns all the parameters used in the function, the weight matrix, the labels for the classification, the number of classes found, the bias, the beta activation function and the accuracy on the trainingset

**References**


**Examples**

```r
x = runif(100, 0, 50)
y = sqrt(x)
train = data.frame(y,x)
train = data.frame(preProcess(train))
OSelm_train.formula(y~x, train, "regression", 100, "hardlim", 10, 10)
```

**predict_elm**

**Prediction function for the ELM model generated with the elm_training() function**

**Description**

Prediction function for the ELM model generated with the elm_training() function

**Usage**

```r
predict_elm(model, test)
```

**Arguments**

- **model**: the output of the elm_training() function
- **test**: dataset used to perform the testing of the model, the first column must be the column to be fitted for the regression or the labels for the classification
preProcess

Value

returns the accuracy on the testset

References


Examples

```r
x = runif(100, 0, 50)
y = sqrt(x)
train = data.frame(y, x)
train = data.frame(preProcess(train))
model = OSelm_train.formula(y~x, train, "regression", 100, "hardlim", 10, 10)
# x = runif(100, 0, 50)
y = sqrt(x)
test = data.frame(y, x)
test = data.frame(preProcess(train))
accuracy = predict_elm(model, test)
```

Description

Pre processing function for the training and test data set. Each numeric variable is standardized between -1 and 1 and each categorical variable is coded with a dummy coding.

Usage

```r
preProcess(data)
```

Arguments

data to be preprocessed

Value

return the pre processed dataset
Index

OSelm_train.formula, 2
OSelm_training, 2

predict(elm, 3
preProcess, 4