Package: ICompELM (via r-universe)

October 31, 2024

Type Package

Title Independent Component Analysis Based Extreme Learning Machine

Version 0.1.0

Description Single Layer Feed-forward Neural networks (SLFNs) have many applications in various fields of statistical modelling, especially for time-series forecasting. However, there are some major disadvantages of training such networks via the widely accepted 'gradient-based backpropagation' algorithm, such as convergence to local minima, dependencies on learning rate and large training time. These concerns were addressed by Huang et al. (2006) <doi:10.1016/j.neucom.2005.12.126>, wherein they introduced the Extreme Learning Machine (ELM), an extremely fast learning algorithm for SLFNs which randomly chooses the weights connecting input and hidden nodes and analytically determines the output weights of SLFNs. It shows good generalized performance, but is still subject to a high degree of randomness. To mitigate this issue, this package uses a dimensionality reduction technique given in Hyvarinen (1999) <doi:10.1109/72.761722>, namely, the Independent Component Analysis (ICA) to determine the input-hidden connections and thus, remove any sort of randomness from the algorithm. This leads to a robust, fast and stable ELM model. Using functions within this package, the proposed model can also be compared with an existing alternative based on the Principal Component Analysis (PCA) algorithm given by Pearson (1901) <doi:10.1080/14786440109462720>, i.e., the PCA based ELM model given by Castano et al. (2013) <doi:10.1007/s11063-012-9253-x>, from which the implemented ICA based algorithm is greatly inspired.

Imports stats, tsutils, ica

Suggests forecast

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Author Saikath Das [aut, cre], Ranjit Kumar Paul [aut], Md Yeasin [aut], Amrit Kumar Paul [aut]

Maintainer Saikath Das <saikathdas007@gmail.com>

Date/Publication 2024-06-10 17:00:14 UTC

Repository https://saikathd.r-universe.dev

RemoteUrl https://github.com/cran/ICompELM

RemoteRef HEAD

RemoteSha f4ce1eed58cb94d4e4f359824ae1fbe64cc87ec4

Contents

`ica.elm_forecast` *Forecasting from ICA based ELM model*

Description

Forecasts are generated recursively from a trained Extreme Learning Machine built using Independent Component Analysis.

Usage

```
ica.elm_forecast(ica.elm_model, h = 1)
```

Arguments

`ica.elm_model`	A trained ICA based ELM model.
`h`	Number of periods for forecasting. Defaults to one-step ahead forecast.

Value

Vector of point forecasts.

See Also

ica.elm_train() for training an ICA based ELM model.

Examples

```
train_set <- head(price, 12*12)
test_set <- tail(price, 12)
ica.model <- ica.elm_train(train_data = train_set, lags = 12)
y_hat <- ica.elm_forecast(ica.elm_model = ica.model, h = length(test_set))
# Evaluation of the forecasts
if(require("forecast")) forecast::accuracy(y_hat, test_set)
```

---

ica.elm_train *Training of ICA based ELM model for time series forecasting*

Description

An Extreme Learning Machine is trained by utilizing the concept of Independent Component Analysis.

Usage

```
ica.elm_train(train_data, lags, comps = lags, bias = TRUE, actfun = "sig")
```

Arguments

train_data	A univariate time series data.
lags	Number of lags to be considered.
comps	Number of independent components to be considered. Corresponds to number of hidden nodes. Defaults to maximum value, i.e., lags.
bias	Whether to include bias term while computing output weights. Defaults to TRUE.
actfun	Activation function for the hidden layer. Defaults to sig. See Activation functions.

Details

An Extreme Learning Machine (ELM) is trained wherein the weights connecting the input layer and hidden layer are obtained using Independent Component Analysis (ICA), instead of being chosen randomly. The number of hidden nodes is determined by the number of independent components.

Value

A list containing the trained ICA-ELM model with the following components.

inp_weights	Weights connecting the input layer to hidden layer, obtained from the unmixing matrix W of ICA. The columns represent the hidden nodes while rows represent input nodes.
out_weights	Weights connecting the hidden layer to output layer.

`fitted.values`	Fitted values of the model.
`residuals`	Residuals of the model.
`h.out`	A data frame containing the hidden layer outputs (activation function applied) with columns representing hidden nodes and rows representing observations.
`data`	The univariate `ts` data used for training the model.
`lags`	Number of lags used during training.
`comps`	Number of independent components considered for training. It determines the number of hidden nodes.
`bias`	Whether bias node was included during training.
`actfun`	Activation function for the hidden layer. See `Activation functions`.

Activation functions

The activation function for the hidden layer must be one of the following.

`sig` Sigmoid function: $(1 + e^{-x})^{-1}$

`radbas` Radial basis function: e^{-x^2}

`hardlim` Hard-limit function: $\begin{cases} 1, & if\ x \geq 0 \\ 0, & if\ x < 0 \end{cases}$

`hardlims` Symmetric hard-limit function: $\begin{cases} 1, & if\ x \geq 0 \\ -1, & if\ x < 0 \end{cases}$

`satlins` Symmetric saturating linear function: $\begin{cases} 1, & if\ x \geq 1 \\ x, & if\ -1 < x < 1 \\ -1, & if\ x \leq -1 \end{cases}$

`tansig` Tan-sigmoid function: $2(1 + e^{-2x})^{-1} - 1$

`tribas` Triangular basis function: $\begin{cases} 1 - |x|, & if\ -1 \leq x \leq 1 \\ 0, & otherwise \end{cases}$

`poslin` Postive linear function: $\begin{cases} x, & if\ x \geq 0 \\ 0, & otherwise \end{cases}$

References

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: theory and applications. Neurocomputing, 70(1-3), 489-501. doi:10.1016/j.neucom.2005.12.126.

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. IEEE transactions on Neural Networks, 10(3), 626-634. doi:10.1109/72.761722.

See Also

`ica.elm_forecast()` for forecasting from trained ICA based ELM model.

Examples

```
train_set <- head(price, 12*12)
ica.model <- ica.elm_train(train_data = train_set, lags = 12)
```

`pca.elm_forecast` *Forecasting from PCA based ELM model*

Description

Forecasts are generated recursively from a trained Extreme Learning Machine built using Principal Component Analysis.

Usage

```
pca.elm_forecast(pca.elm_model, h = 1)
```

Arguments

`pca.elm_model`	A trained PCA based ELM model.
`h`	Number of periods for forecasting. Defaults to one-step ahead forecast.

Value

Vector of point forecasts.

See Also

`pca.elm_train()` for training an ICA based ELM model.

Examples

```
train_set <- head(price, 12*12)
test_set <- tail(price, 12)
pca.model <- pca.elm_train(train_data = train_set, lags = 12)
y_hat <- pca.elm_forecast(pca.elm_model = pca.model, h = length(test_set))
# Evaluation of the forecasts
if(require("forecast")) forecast::accuracy(y_hat, test_set)
```

`pca.elm_train` *Training of PCA based ELM model for time series forecasting*

Description

An Extreme Learning Machine is trained by utilizing the concept of Principal Component Analysis.

Usage

```
pca.elm_train(
  train_data,
  lags,
  comps = lags,
  center = TRUE,
  scale = TRUE,
  bias = TRUE,
  actfun = "sig"
)
```

Arguments

`train_data`	A univariate time series data.
`lags`	Number of lags to be considered.
`comps`	Number of independent components to be considered. Corresponds to number of hidden nodes. Defaults to maximum value, i.e., `lags`.
`center`	Whether to compute PCA on mean-adjusted data.
`scale`	Whether to compute PCA on variance-adjusted data.
`bias`	Whether to include bias term while computing output weights. Defaults to `TRUE`.
`actfun`	Activation function for the hidden layer. Defaults to `sig`. See `Activation functions`.

Details

An Extreme Learning Machine (ELM) is trained wherein the weights connecting the input layer and hidden layer are obtained using Principal Component Analysis (PCA), instead of being chosen randomly. The number of hidden nodes is determined by the number of principal components.

Value

A list containing the trained ICA-ELM model with the following components.

`inp_weights`	Weights connecting the input layer to hidden layer, obtained from the unmixing matrix W of ICA. The columns represent the hidden nodes while rows represent input nodes.
`out_weights`	Weights connecting the hidden layer to output layer.
`fitted.values`	Fitted values of the model.

`residuals`	Residuals of the model.
`h.out`	A data frame containing the hidden layer outputs (activation function applied) with columns representing hidden nodes and rows representing observations.
`data`	The univariate `ts` data used for training the model.
`lags`	Number of lags used during training.
`comps`	Number of independent components considered for training. It determines the number of hidden nodes.
`center`	Whether the input data was mean-adjusted during training.
`scale`	Whether the input data was variance-adjusted during training.
`bias`	Whether bias node was included during training.
`actfun`	Activation function for the hidden layer. See `Activation functions`.

Activation functions

The activation function for the hidden layer must be one of the following.

`sig` Sigmoid function: $(1 + e^{-x})^{-1}$

`radbas` Radial basis function: e^{-x^2}

`hardlim` Hard-limit function: $\begin{cases} 1, & if\ x \geq 0 \\ 0, & if\ x < 0 \end{cases}$

`hardlims` Symmetric hard-limit function: $\begin{cases} 1, & if\ x \geq 0 \\ -1, & if\ x < 0 \end{cases}$

`satlins` Symmetric saturating linear function: $\begin{cases} 1, & if\ x \geq 1 \\ x, & if\ -1 < x < 1 \\ -1, & if\ x \leq -1 \end{cases}$

`tansig` Tan-sigmoid function: $2(1 + e^{-2x})^{-1} - 1$

`tribas` Triangular basis function: $\begin{cases} 1 - |x|, & if\ -1 \leq x \leq 1 \\ 0, & otherwise \end{cases}$

`poslin` Postive linear function: $\begin{cases} x, & if\ x \geq 0 \\ 0, & otherwise \end{cases}$

References

Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science, 2(11), 559-572. doi:10.1080/14786440109462720.

Castaño, A., Fernández-Navarro, F., & Hervás-Martínez, C. (2013). PCA-ELM: a robust and pruned extreme learning machine approach based on principal component analysis. Neural processing letters, 37, 377-392. doi:10.1007/s11063-012-9253-x.

See Also

`pca.elm_forecast()` for forecasing from trained PCA based ELM model.

Examples

```
train_set <- head(price, 12*12)
pca.model <- pca.elm_train(train_data = train_set, lags = 12)
```

`price` *Aggregate gram price data*

Description

National aggregate price of gram from Indian markets, which is a major pulse in the country. The observations range from January, 2010 upto December, 2023.

Usage

```
price
```

Format

A `ts` object with 156 observations.

Source

https://www.agmarknet.gov.in/

Examples

```
plot(price, xlab = "Year", ylab = "Aggregate price of Gram (Rs./Bag)")
```

Index