Package 'knnp'

July 1, 2018

Version 1.0.0

Date 2018-06-18

Title Time Series Prediction using K-Nearest Neighbors Algorithm (Parallel)

Depends R (>= 3.3.3)

Imports parallelDist, forecast, stats, utils, doParallel, foreach

Description Two main functionalities are provided. One of them is predicting values with k-nearest neighbors algorithm and the other is optimizing the parameters k and d of the algorithm. These are carried out in parallel using multiple threads.

License AGPL-3

RoxygenNote 6.0.1

URL https://github.com/Dani-Basta/TFG

BugReports https://github.com/Dani-Basta/TFG/issues

NeedsCompilation no

Author Daniel Bastarrica Lacalle [aut], Javier Berdecio Trigueros [aut, cre]

Maintainer Javier Berdecio Trigueros <javierberdeciot@gmail.com>

Repository CRAN

Date/Publication 2018-07-01 15:00:02 UTC

R topics documented:

knn_distances 2
knn_elements 3
knn_next 3
knn_optim 4
knn_optim_parallel 6
knn_optim_parallel2 7
knn_optim_parallelf 8
knn_past 10

Index **12**

knn_distances *Distances matrixes computation and saving in files with a maximum of columns*

Description

Calculates one distances matrix per each d for the given time series and then save them in files. Each file will contain a maximum of 'cols' number of columns from the corresponding distances matrix.

Usage

```
knn_distances(y, d, distance_metric = "euclidean", threads = NULL, file,
  cols = 1)
```

Arguments

y	A time series.
d	Values of d's to be analyzed.
distance_metric	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, canberra and others. For more information about the supported metrics check the values that 'method' argument of function 'parDist' (from 'parallelDist' package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/package=parallelDist. Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "canberra", "chord".
threads	Number of threads to be used when parallelizing distances calculation, default is number of cores detected - 1 or 1 if there is only one core.
file	Path and id of the files where the distances matrixes will be saved.
cols	Number of columns per file.

Examples

```
knn_distances(AirPassengers, 1:3, threads = 2, file = "AirPassengers", cols = 2)
knn_distances(LakeHuron, 1:6, threads = 2, file = "LakeHuron", cols = 10)
```

knn_elements *'Elements' matrix computation*

Description

Creates a matrix to be used for calculating distances. The most recent 'element' is put in the first row of the matrix, the second most recent 'element' in the second row and so on. Therefore, the oldest 'element' is put in the last row.

Usage

```
knn_elements(y, d)
```

Arguments

y	A matrix.
d	Length of each of the 'elements'.

Value

A matrix to be used for calculating distances.

knn_next *Next value prediction*

Description

Predicts next value of the time series using k-nearest neighbors algorithm.

Usage

```
knn_next(y, k, d, v = 1, distance_metric = "euclidean",
  weight = "proximity", threads = NULL)
```

Arguments

y	A time series.
k	Number of neighbors.
d	Length of each of the 'elements'.
v	Variable to be predicted if given multivariate time series.

`distance_metric`
: Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, canberra and others. For more information about the supported metrics check the values that 'method' argument of function 'parDist' (from 'parallelDist' package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/package=parallelDist. Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "canberra", "chord".

`weight`
: Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proximity, same, linear.

  proximity the weight assigned to each neighbor is proportional to its distance

  same all neighbors are assigned with the same weight

  linear nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.

`threads`
: Number of threads to be used when parallelizing distances calculation, default is number of cores detected - 1 or 1 if there is only one core.

Value

The predicted value.

Examples

```
knn_next(AirPassengers, 5, 2, threads = 2)
knn_next(LakeHuron, 3, 6, threads = 2)
```

---

knn_optim *k and d optimization*

Description

Optimizes the values of k and d for a given time series. First, values corresponding to instants from init + 1 to the last one are predicted. The first value predicted, which corresponds to instant init + 1, is calculated using instants from 1 to instant init; the second value predicted, which corresponds to instant init + 2, is predicted using instants from 1 to instant init + 1; and so on until the last value, which corresponds to instant n (length of the given time series), is predicted using instants from 1 to instant n - 1. Finally, the error is evaluated between the predicted values and the real values of the series. This version of the optimization function only uses one thread except for the distances matrixes calculation, for which the number of threads to be used can be specified.

Usage

```
knn_optim(y, k, d, v = 1, init = NULL, distance_metric = "euclidean",
  error_metric = "MAE", weight = "proximity", threads = NULL)
```

Arguments

y	A time series.
k	Values of k's to be analyzed.
d	Values of d's to be analyzed.
v	Variable to be predicted if given multivariate time series.
init	Variable that determines the limit of the known past for the first instant predicted.
distance_metric	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, canberra and others. For more information about the supported metrics check the values that 'method' argument of function 'parDist' (from 'parallelDist' package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/package=parallelDist. Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "canberra", "chord".
error_metric	Type of metric to evaluate the prediction error. Five metrics supported: **ME** Mean Error **RMSE** Root Mean Squared Error **MAE** Mean Absolute Error **MPE** Mean Percentage Error **MAPE** Mean Absolute Percentage Error
weight	Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proximity, same, linear. **proximity** the weight assigned to each neighbor is proportional to its distance **same** all neighbors are assigned with the same weight **linear** nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.
threads	Number of threads to be used when parallelizing, default is number of cores detected - 1 or 1 if there is only one core.

Value

A matrix of errors, optimal k and d.

Examples

```
knn_optim(AirPassengers, 1:5, 1:3, threads = 2)
knn_optim(LakeHuron, 1:10, 1:6, threads = 2)
```

`knn_optim_parallel` *Parallel k and d optimization*

Description

Optimizes the values of K and D for a given time series. First, values corresponding to instants from init + 1 to the last one are predicted. The first value predicted, which corresponds to instant init + 1, is calculated using instants from 1 to instant init; the second value predicted, which corresponds to instant init + 2, is predicted using instants from 1 to instant init + 1; and so on until the last value, which corresponds to instant n (length of the given time series), is predicted using instants from 1 to instant n - 1. Finally, the error is evaluated between the predicted values and the real values of the series. This version of the optimization function uses a parallelized distances calculation function, and the computation of the predicted values is done parallelizing by the number of d's and the number of instants to be predicted.

Usage

```
knn_optim_parallel(y, k, d, v = 1, init = NULL,
  distance_metric = "euclidean", error_metric = "MAE",
  weight = "proximity", threads = NULL)
```

Arguments

`y`	A time series.
`k`	Values of k's to be analyzed.
`d`	Values of d's to be analyzed.
`v`	Variable to be predicted if given multivariate time series.
`init`	Variable that determines the limit of the known past for the first instant predicted.
`distance_metric`	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, canberra and others. For more information about the supported metrics check the values that 'method' argument of function 'parDist' (from 'parallelDist' package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/package=parallelDist. Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "canberra", "chord".
`error_metric`	Type of metric to evaluate the prediction error. Five metrics supported: **ME** Mean Error **RMSE** Root Mean Squared Error **MAE** Mean Absolute Error **MPE** Mean Percentage Error **MAPE** Mean Absolute Percentage Error
`weight`	Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proximity, same, linear.

proximity the weight assigned to each neighbor is proportional to its distance

same all neighbors are assigned with the same weight

linear nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.

`threads` Number of threads to be used when parallelizing, default is number of cores detected - 1 or 1 if there is only one core.

Value

A matrix of errors, optimal k and d.

Examples

```
knn_optim_parallel(AirPassengers, 1:5, 1:3, threads = 2)
knn_optim_parallel(LakeHuron, 1:10, 1:6, threads = 2)
```

---

knn_optim_parallel2 *Parallel k and d optimization*

Description

Optimizes the values of k and d for a given time series. First, values corresponding to instants from init + 1 to the last one are predicted. The first value predicted, which corresponds to instant init + 1, is calculated using instants from 1 to instant init; the second value predicted, which corresponds to instant init + 2, is predicted using instants from 1 to instant init + 1; and so on until the last value, which corresponds to instant n (length of the given time series), is predicted using instants from 1 to instant n - 1. Finally, the error is evaluated between the predicted values and the real values of the series. This version of the optimization function uses a parallelized distances calculation function, and the computation of the predicted values is done parallelizing by the number of d's.

Usage

```
knn_optim_parallel2(y, k, d, v = 1, init = NULL,
  distance_metric = "euclidean", error_metric = "MAE",
  weight = "proximity", threads = NULL)
```

Arguments

`y`	A time series.
`k`	Values of k's to be analyzed.
`d`	Values of d's to be analyzed.
`v`	Variable to be predicted if given multivariate time series.
`init`	Variable that determines the limit of the known past for the first instant predicted.

`distance_metric`
: Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, canberra and others. For more information about the supported metrics check the values that 'method' argument of function 'parDist' (from 'parallelDist' package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/package=parallelDist. Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "canberra", "chord".

`error_metric`
: Type of metric to evaluate the prediction error. Five metrics supported:

  ME Mean Error

  RMSE Root Mean Squared Error

  MAE Mean Absolute Error

  MPE Mean Percentage Error

  MAPE Mean Absolute Percentage Error

`weight`
: Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proximity, same, linear.

  proximity the weight assigned to each neighbor is proportional to its distance

  same all neighbors are assigned with the same weight

  linear nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.

`threads`
: Number of threads to be used when parallelizing, default is number of cores detected - 1 or 1 if there is only one core.

Value

A matrix of errors, optimal k and d.

Examples

```
knn_optim_parallel2(AirPassengers, 1:5, 1:3, threads = 2)
knn_optim_parallel2(LakeHuron, 1:10, 1:6, threads = 2)
```

---

knn_optim_parallelf *Parallel k and d optimization reading from files*

Description

Optimizes the values of k and d for a given time series. First, values corresponding to instants from init + 1 to the last one are predicted. The first value predicted, which corresponds to instant init + 1, is calculated using instants from 1 to instant init; the second value predicted, which corresponds to instant init + 2, is predicted using instants from 1 to instant init + 1; and so on until the last value, which corresponds to instant n (length of the given time series), is predicted using instants from 1 to instant n - 1. Finally, the error is evaluated between the predicted values and the real values of the series. This version of the optimization function uses a parallelized distances calculation

function, and the computation of the predicted values is done parallelizing by the number of d's and the number of instants to be predicted. Each thread that calculates predicted values reads only the part of the corresponding distances matrix in which the information used to predict is contained.

Usage

```
knn_optim_parallelf(y, k, d, v = 1, init = NULL, error_metric = "MAE",
  weight = "proximity", threads = NULL, file, cols)
```

Arguments

y	A time series.
k	Values of k;s to be analyzed.
d	Values of d's to be analyzed.
v	Variable to be predicted if given multivariate time series.
init	Variable that determines the limit of the known past for the first instant predicted.
error_metric	Type of metric to evaluate the prediction error. Five metrics supported: **ME** Mean Error **RMSE** Root Mean Squared Error **MAE** Mean Absolute Error **MPE** Mean Percentage Error **MAPE** Mean Absolute Percentage Error
weight	Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proximity, same, linear. **proximity** the weight assigned to each neighbor is proportional to its distance **same** all neighbors are assigned with the same weight **linear** nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.
threads	Number of threads to be used when parallelizing, default is number of cores detected - 1 or 1 if there is only one core.
file	Path and id of the files where the distances matrixes are.
cols	Number of columns per file.

Value

A matrix of errors, optimal k and d.

Examples

```
knn_distances(AirPassengers, 1:3, file = "AirPassengers", cols = 2, threads = 2)
knn_optim_parallelf(AirPassengers, 1:5, 1:3, file = "AirPassengers", cols = 2, threads = 2)
knn_distances(LakeHuron, 1:6, file = "LakeHuron", cols = 10, threads = 2)
knn_optim_parallelf(LakeHuron, 1:10, 1:6, file = "LakeHuron", cols = 10, threads = 2)
```

knn_past *Past time prediction*

Description

Predicts values of the time series using k-nearest neighbors algorithm. Values corresponding to instants from init + 1 to the last one are predicted. The first value predicted, which corresponds to instant init + 1, is calculated using instants from 1 to instant init; the second value predicted, which corresponds to instant init + 2, is predicted using instants from 1 to instant init + 1; and so on until the last value, which corresponds to instant n (length of the given time series), is predicted using instants from 1 to instant n - 1.

Usage

```
knn_past(y, k, d, v = 1, init = NULL, distance_metric = "euclidean",
  weight = "proximity", threads = NULL)
```

Arguments

y	A time series.
k	Number of neighbors.
d	Length of each of the 'elements'.
v	Variable to be predicted if given multivariate time series.
init	Variable that determines the limit of the known past for the first instant predicted.
distance_metric	Type of metric to evaluate the distance between points. Many metrics are supported: euclidean, manhattan, dynamic time warping, canberra and others. For more information about the supported metrics check the values that 'method' argument of function 'parDist' (from 'parallelDist' package) can take as this is the function used to calculate the distances. Link to the package info: https://cran.r-project.org/package=parallelDist. Some of the values that this argument can take are "euclidean", "manhattan", "dtw", "canberra", "chord".
weight	Type of weight to be used at the time of calculating the predicted value with a weighted mean. Three supported: proximity, same, linear. **proximity** the weight assigned to each neighbor is proportional to its distance **same** all neighbors are assigned with the same weight **linear** nearest neighbor is assigned with weight k, second closest neighbor with weight k-1, and so on until the least nearest neighbor which is assigned with a weight of 1.
threads	Number of threads to be used when parallelizing, default is number of cores detected - 1 or 1 if there is only one core.

Value

The predicted values.

Examples

```
knn_past(AirPassengers, 5, 2, threads = 2)
knn_past(LakeHuron, 3, 6, threads = 2)
```

Index

knn_distances, 2
knn_elements, 3
knn_next, 3
knn_optim, 4
knn_optim_parallel, 6
knn_optim_parallel2, 7
knn_optim_parallelf, 8
knn_past, 10