Package: LCCkNN (via r-universe)

May 26, 2026

Title Adaptive k-Nearest Neighbor Classifier Based on Local Curvature Estimation

Version 0.1.0

Description Implements the kK-NN algorithm, an adaptive k-nearest neighbor classifier that adjusts the neighborhood size based on local data curvature. The method estimates local Gaussian curvature by approximating the shape operator of the data manifold. This approach aims to improve classification performance, particularly in datasets with limited samples.

License MIT + file LICENSE

URL https://github.com/Gabrielforest/LCCkNN

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports FNN, caret, MLmetrics, stats, class

RoxygenNote 7.3.2

Config/pak/sysreqs libicu-dev

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

Date/Publication 2025-08-27 12:46:06 UTC

RemoteUrl https://github.com/gabrielforest/lccknn

RemoteRef HEAD

RemoteSha 711eae6ed089108293d9c1b30b93853aa574c60a

Contents

`balanced_accuracy_score` *Computes balanced accuracy.*

Description

This function requires the 'caret' package.

Usage

```
balanced_accuracy_score(true_labels, predicted_labels)
```

Arguments

`true_labels`	The true class labels.
`predicted_labels`	The predicted class labels.

Value

The balanced accuracy score.

`curvature_estimation` *Computes the curvatures of all samples in the training set.*

Description

Computes the curvatures of all samples in the training set.

Usage

```
curvature_estimation(data, k)
```

Arguments

`data`	A numeric matrix or data frame of the training data.
`k`	The number of neighbors for the initial k-NN graph.

Value

A numeric vector of curvatures for each sample.

f1_score *Computes the F1-score.*

Description

Computes the F1-score.

Usage

```
f1_score(true_labels, predicted_labels, average = "weighted")
```

Arguments

true_labels	The true class labels.
predicted_labels	The predicted class labels.
average	The type of averaging ('weighted').

Value

The F1-score.

kKNN *Adaptive k-Nearest Neighbor Classifier*

Description

Implements the adaptive k-nearest neighbor (kK-NN) algorithm, which adjusts the neighborhood size for each sample based on a local curvature estimate. This method aims to improve classification performance, particularly in datasets with limited training samples.

Usage

```
kKNN(train, test, train_target, k, func = "log", quantize_method = "paper")
```

Arguments

train	A numeric matrix or data frame of the training data.
test	A numeric matrix or data frame of the test data.
train_target	A numeric or factor vector of class labels for the training data.
k	The number of neighbors for the initial k-NN graph.
func	The transformation function for curvatures ('log', 'cubic_root', or 'sigmoid').
quantize_method	The quantization method to use: 'paper' (10 levels, default) or 'log2n' (k levels, where k = log2(n)).

Value

A numeric or factor vector of predicted class labels for the test data.

References

Levada, A.L.M., Nielsen, F., Haddad, M.F.C. (2024). ADAPTIVE k-NEAREST NEIGHBOR CLASSIFIER BASED ON THE LOCAL ESTIMATION OF THE SHAPE OPERATOR. arXiv:2409.05084.

Examples

```
# Load necessary libraries
library(caret)

# Load and prepare data (e.g., the Iris dataset)
data_iris <- iris
data <- as.matrix(data_iris[, 1:4])
target <- as.integer(data_iris$Species)

# Standardize the data
data <- scale(data)

# Split data into training and testing sets
set.seed(42)
train_index <- caret::createDataPartition(target, p = 0.5, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
train_labels <- target[train_index]

# Determine initial k value as log2(n)
initial_k <- round(log2(nrow(train_data)))
if (initial_k %% 2 == 0) {
   initial_k <- initial_k + 1
}

# Run the kK-NN classifier using the default quantization method ('paper')
predictions_paper <- LCCkNN::kKNN(
   train = train_data,
   test = test_data,
   train_target = train_labels,
   k = initial_k
)

# Run the kK-NN classifier using the 'log2n' quantization method
predictions_log2n <- LCCkNN::kKNN(
   train = train_data,
   test = test_data,
   train_target = train_labels,
   k = initial_k,
   quantize_method = 'log2n'
)

# Evaluate the results (e.g., calculate balanced accuracy)
```

```
test_labels <- target[-train_index]
bal_acc_paper <- LCCkNN::balanced_accuracy_score(test_labels, predictions_paper)
bal_acc_log2n <- LCCkNN::balanced_accuracy_score(test_labels, predictions_log2n)
cat("Balanced Accuracy (paper Method):", bal_acc_paper, "\n")
cat("Balanced Accuracy (log2n Method):", bal_acc_log2n, "\n")
```

`point_curvature_estimation` *Computes the curvature of a single test sample's neighborhood.*

Description

Computes the curvature of a single test sample's neighborhood.

Usage

```
point_curvature_estimation(data)
```

Arguments

`data`	A numeric matrix or data frame representing the neighborhood (test point + its neighbors).

Value

A single numeric value for the curvature.

`quantize` *Quantizes real values to integer levels.*

Description

This function quantizes real values in the interval `[a, b]` to integer levels from 0 to k-1.

Usage

```
quantize(arr, a, b, k = 10)
```

Arguments

`arr`	A numeric vector in the interval `[a, b]`.
`a`	The lower bound of the interval.
`b`	The upper bound of the interval.
`k`	The number of quantization levels (default is 10).

Value

A vector of quantized integers in `0, ..., k - 1`.

`sigmoid` *A helper sigmoid function.*

Description

A helper sigmoid function.

Usage

```
sigmoid(x, a = 1)
```

Arguments

`x`	A numeric value or vector.
`a`	A numeric scaling factor (default is 1).

Value

The sigmoid of x.

`testa_KNN` *Standard k-NN classifier.*

Description

Standard k-NN classifier.

Usage

```
testa_KNN(train, test, train_target, nn)
```

Arguments

`train`	A numeric matrix or data frame of the training data.
`test`	A numeric matrix or data frame of the test data.
`train_target`	A numeric or factor vector of class labels for the training data.
`nn`	The number of neighbors.

Value

A numeric or factor vector of predicted class labels.

Index